AD-A118 469 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/2
PRODUCTION SYSTEMS AS A PROGRAMMING LANGUAGE FOR ARTIFICIAL INT--ETC(U)
DEC 76 M D RYCHENER. F44620-73-C-0074

UNCLASSIFIED NL

1 of 2
AD
A118469

Production Systems as a Programming Language for Artificial Intelligence Applications

Volume III

Michael D. Rychener

December 1976

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa 15213

This report reproduces a dissertation submitted to the Department of Computer Science at Carnegie-Mellon University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

Table of Contents

For Volume III

Preface to Volume III

This volume contains two chapters, covering work with production systems in the areas of natural language processing and game playing. Chapter V describes a program that plays a simple class of chess endgames, and discusses the possibilities of using production systems for chess in general. Chapter VI describes a system that carries on a dialog about a toy blocks world, and that solves a class of problems in that world similar to the capabilities of Winograd's system. Each chapter has an abstract and a detailed table of contents. It is assumed that the reader has some familiarity with Volume I of this report, which discusses the goals and conclusions of the thesis as a whole, and which introduces the production system language in which the systems in this volume are implemented. The chapters have a similar organization, starting with a general description of the task performed by the system, and proceeding to a description of the system and its behavior. There are sections that discuss issues with respect to the task itself and with respect to the use of production systems.

Chapter V

KPKEG

A Production System for King-Pawn-King Chess Endgames

Abstract. KPKEG is a production system implementation of a program that plays chess endgames, restricted to king and pawn versus king. The program is described and several examples of its operation are discussed. The program's chess knowledge is given, and how this knowledge is expressed as productions is described. Experiments with KPKEG have brought out several features of the principle on which the search is based and the chess knowledge organized, the strategy hierarchy. Features of the productions and how they compare with a Lisp version of a similar program bring out the advantages of this implementation. The productions lend themselves readily to extension to more demanding chess tasks.

Table of Contents

For Chapter V

A. Introduction

This chapter is concerned with a PS program, called KPKEG (king pawn king endgame), for playing a restricted form of chess, namely, chess endgames in which a king and a lone pawn of one color are opposed by a lone king (hereafter, the subset of chess with king and pawn versus king will be abbreviated K-P-K). Although chess is a specialized area of AI, and is probably a suitable domain only for chess experts (which I am not), it will still be useful for the present thesis for the following reasons. As Berliner (1973) has argued, the classical heuristic search approach to chess has fundamental limitations, which have been observed empirically in performance and theoretically by Berliner on the basis of critical situations in which the search techniques appear to be hopelessly inappropriate. Consequently there has been a shift in emphasis towards bringing to bear more of the kinds of chess knowledge used by human players. Since PSs are being put forth as useful tools in encoding problem-solving knowledge, it is reasonable to do preliminary experiments along the lines here. In addition, even a restricted chess program provides an easy benchmark for comparison with other control structures, since a variety of other programs exist, with a current effort using Lisp on a very similar chess domain.

The central chess concepts behind KPKEG are provided by Fine's (1941) analysis of K-P-K endgames. In this problem area there are a reasonably small number of pieces of knowledge that prove to be adequate for correct analysis. That is, KPKEG relies heavily on the use of patterns of chess pieces and much less on a search of possible move sequences leading to a win or draw. Patterns are used both to direct the program's attention to effective moves and to evaluate positions reached by the search. The search of possible variations of play is conducted under an executive scheme called a strategy hierarchy, developed by Berliner (1975b) as appropriate (at least) to the K-P-K domain. The strategy hierarchy in KPKEG consists of seven levels (to be described in more detail presently), each of which has associated with it goals and move-generation procedures for attempting to achieve the goals. The principle for constructing the hierarchy is that a lower strategy is never attempted in refuting a higher one. On the other hand, a move that attains the goal of a higher strategy is a good refutation of a move aimed at attaining the goal of a lower strategy, since a higher strategy is globally more valuable in the sense that it is more essential to achieving the best game outcome. The way that the hierarchy is used to generate a search tree of moves and replies is that at the top level a player starts by trying to achieve his highest strategy. When that fails, he decreases his strategy level and tries to achieve a success at that level. The opponent, who moves at a lower depth, tries to refute the top strategy by first trying to achieve a strategy at the same level, and then when that fails, by trying moves at higher strategic levels. The search tree is generated as the players alternate in trying to refute plays at higher depths, until a position known to be a win or a draw occurs.

This chapter first presents KPKEG in detail, Section B, and describes several experiments that exhibit its behavior, Section C. Specific issues with respect to PSs are discussed in Section D. In Section E, KPKEG is compared to a similar program implemented in Lisp, with particular attention to the use of PSs to achieve the control structures of the Lisp version. Finally, we consider whether KPKEG can serve as a solid basis for further research, which is important because of the limited aims of the present work.

B. The KPKEG Program in Detail

The objectives of either player in a chess game with only two kings and a pawn are limited. The player with only a king must achieve stalemate, capture the pawn, or block it from its promotion square, in order to obtain a draw. The player with the pawn must promote it safely while avoiding stalemate. To achieve these overall strategic objectives there are a number of lesser considerations, such as controlling the square in front of the pawn, forcing the enemy king in some direction, gaining the opposition (a chess term to be defined below), and advancing the pawn. These objectives have been formulated in KPKEG into seven levels, each assigned a corresponding numerical value.

7	Mate (White) or capture pawn (Black)
6	Queen the pawn or stalemate
5	Advance the pawn or occupy pawn's queening square
4	Control the pawn's path
3	Defend or attack the pawn
2	Restrict (force) the enemy king's move
1	Any other move (essentially away from pawn or enemy king)

The goal for the program then becomes to execute successfully a move at the highest possible strategy level. In this section we first illustrate how such a move is arrived at in a particular example, and then proceed to a more detailed discussion of KPKEG.

B.1. A simple example of program behavior

The position that we will examine is given in Figure B.1, and the complete program behavior trace is given in Appendix D. White starts out trying to achieve its highest strategy, which is to move the pawn onto its queening square; this strategy is at level 6 in KPKEG's hierarchy. The queening square is E8 (using the program's algebraic notation, which is indicated in the figure) and the pawn is at E6, so this fails immediately. White then decreases its level to 5, where the objective is to advance the pawn. Black's level-5 strategy is to intercept the pawn, preventing its advance. White advances the pawn to E7 and black responds C7-D7, at which position the black king is in control of the pawn and its queening square, and the white king is not within striking distance, so that White's advance-pawn strategy can make no further moves. Black's strategy succeeds because White fails to respond, and this success is a refutation of White's top-level move. White has no other ways to implement its level-5 strategy, so advancing the pawn directly is abandoned. White starts over at the initial position with strategy level decreased to 4, whose objective is to control with the king the path of the pawn's advance. Black's corresponding strategy is to occupy the pawn's path to its queening square (which is only slightly different from its level-5 strategy). White now moves its king E4-E5, Black responds C7-D8, White responds E5-D6, and black, D8-E8 (see Figure B.1).

Black's king is now on the pawn's queening square (E8) and it controls the square in front of the pawn (E7), so that White can go no further with its strategy to control the path to E8 from E6. Rather than giving up at this position, White increases its strategy level - Black has succeeded at level 4 but that may not be strong enough to refute some of the higher-level white strategies. White's attempt at level 5, moving the pawn from E6-E7, does in fact lead to a winning position for White, since the black king is forced to move off the queening square, whereupon the white king can move to control it. The way the program actually behaves is that Black's strategies fail to generate any moves (as did

```
8 .. .. .. ..        8 .. ..BK.. ..
7.. BK .. ..         7.. .. .. ..
6 .. ..WP.. ..       6 .. WKWP.. ..
5.. .. .. ..         5.. .. .. ..
4 .. ..WK.. ..       4 .. .. .. ..
3.. .. .. ..         3.. .. .. ..
2 .. .. .. ..        2 .. .. .. ..
1.. .. .. ..         1.. .. .. ..
 A B C D E F G H      A B C D E F G H
```

Figure B.1 Starting and intermediate positions for TEST1; White to move

White's strategy at level 5 in the first segment of the trace) and E6-E7 thus succeeds. Black's move preceding E6-E7 fails, and the search proceeds by examining alternatives at that point.

This has described about two-thirds of the first column of the behavior trace in Appendix D (up to number 11), which is about one-fourth of the complete search that KPKEG does before deciding that White's E4-E5 is a satisfactory move. The primary characteristic of the program's search has been illustrated: it searches in a very restricted fashion according to a predetermined ordering of strategies, evaluating positions in the light of the strategic objectives currently in effect. That is, move attempts are generated only when they are deemed relevant to achieving success of a strategy, and the determination of what strategy is in effect depends on the strategy behind the previous move or on maximizing the outcome of the position at the top level. We now proceed to give more detail on KPKEG's internal structure.

B.2. An overview of the structure of KPKEG

The Ps of KPKEG are divided into six main groups: the strategy executive (Ps whose names start with S); Ps for updating the internal representation of the board (Q Ps); means for implementing move strategies (M Ps); strategies for White, or more generally the player who has the pawn (W Ps); strategies for Black (B Ps); and the initialization for example problems (X Ps). The strategy executive maintains information pertaining to the state of the tree search and the current strategy level. It also includes a set of Ps that recognize various patterns known to be wins or draws. The executive evokes the White and Black strategies, and uses the moves that they generate to carry the tree search forward. It uses the updating Ps to make the transition from one node in the tree to another. The strategy Ps generate move candidates directly or generate more abstract descriptions of what they intend, which are then converted to move candidates by the means Ps. We turn now to a more detailed look at the set of Ps in which most of the program control is embodied, the strategy executive. Incidental details of the other Ps are brought out, but fuller detail is postponed until the following subsection.

The VAPs (very abstract Ps) of Figure B.2 represent the main features of the strategy executive and of the other Ps as they appear to it. As the reader will recall from Chapter IV, underlining is used in VAPs to denote super-conditions and super-actions, which represent sets of condition or action elements, or the condensed result of many P

```
            % SE's: Strategy Executive VAPs; 55 actual Ps %
SE1:  findmove -> initialize & select-strategy-move & check-strategy-result;
SE2:  check-strategy-result & strategies-exhausted & not levels-exhausted
      -> change-strategy-level & select-strategy-move & check-strategy-result;
SE3:  check-strategy-result & strategies-exhausted & levels-exhausted
      -> record-position & succeed-strategy-at-previous-depth;
SE4:  best-move-candidate -> make-move & check-terminal-position & check-move-result;
SE5:  check-terminal-position & not terminal-position & not maximum-depth
      -> select-strategy-move & check-strategy-result;
SE6:  check-terminal-position & terminal-position-pattern -> terminal-win;
SE7:  check-terminal-position & maximum-depth & not terminal-position
      -> static-eval-strategy;
SE8:  terminal-win(self) OR succeed-strategy -> refute-strategy-at-previous-depth;
SE9:  terminal-win(opponent) -> succeed-strategy-at-previous-depth;
SE10: check-move-result & refuted
      -> retract-move & continue-to-try-move-candidates-and-strategies;
SE11: check-move-result & succeed & not depth=1
      -> retract-move & refute-strategy-at-previous-depth;
SE12: check-move-result & succeed & depth=1 -> make-actual-play;
SE13: record-position & position-before-making-successful-move
      -> build-P-to-recognize-as-terminal-position
         & build-P-to-recommend-trying-move-if-position-recurs-at
            -greater-depth-or-at-depth-1;

            % UB's: Updating Board for moves; 19 Ps %
UB1:  make-move & move-type & location's & controls's -> location's & controls's;
UB2:  retract-move & move-type & location's & controls's -> location's & controls's;

            % MMC's: Means to Move Candidates; 18 Ps %
MMC1: means-signal & properties-relevant-to-desired-moves -> move-candidate's;

            % WBS's: White and Black Strategies; 44 Ps %
WBS1: select-strategy-move & board-pattern -> means-signal's OR move-candidate's;
WBS2: select-strategy-move & board-pattern -> succeed-strategy;
WBS3: static-eval-strategy & board-pattern -> terminal-win;

            % TX's: Test Examples; 5 Ps for 3 tests %
TX1:  test-signal -> initialize & controls's & location's;
```

Figure B.2 VAPs for KPKEG

firings. Elements of VAPs that are not underlined correspond to actual program elements, and behave similarly with respect to the way Psnlst considers events to be ordered.

Using the VAPs we can follow the example in Section B.1 in enough detail to see the way the program works. At the beginning of TEST1, the user asserts a signal that fires the equivalent of TX1, which sets up the board situation. Then another user signal, "findmove", fires SE1 which initializes the strategy executive and starts the search process by asserting "select-strategy-move". As discussed above White starts out trying to achieve its highest-level strategy; the level of strategy being sought is set by the initialization in SE1. The VAPs that respond to "select-strategy-move" are the WBS's, which generate move candidates or recognize success based on board patterns. In the present case none of the WBS's fires, since the level 6 strategy for White is to move its pawn onto the eighth rank. Nothing responds to "select-strategy-move", so that the "check-strategy-result" signal from SE1 is examined, according to the conditions in SE2 and SE3. The situation is that the strategies at level 6 are exhausted but that the other levels haven't been tried yet so that SE2 is true, causing the level to be decremented to 5 and again asserting the "select-strategy-move" signal.

This time the strategy is to advance the pawn, and a move-candidate (E6-E7) is asserted by an instance of WBS1. SE4 represents the selection of a move-candidate from a set of them. UB1 responds to the "make-move" signal from SE4, updating the board according to the nature of the move (i.e., whether a king or pawn is moving, and the direction of the move). The "check-terminal-position" from SE4 evokes the testing of the patterns represented by SE5, SE6, and SE7; those are patterns for the small number of known won or drawn positions for K-P-K endgames. In the present example, SE5 is appropriate, and sets up the strategy selection for Black, who must respond to White's pawn advance. The program goes through the sequence represented by WBS1, SE4, UB1, and SE5. Black has moved its king to D7, and White's advance-pawn and queen-pawn strategies (instances of WBS1) recognize that further moves are no good, so that SE2 fires, and then SE3 becomes true when no response to "select-strategy-move" is made. Notice that only strategies not less than level 5 are considered by White, in accord with the strategy hierarchy principle - trying a weaker strategy to refute a stronger one makes no sense.

SE3 first causes the position before Black's move to D7 to be recorded as a known success (via SE13). Then the strategy at the previous depth, namely the one that proposed the move to D7, is made to succeed. The success is noted by SE11, which uses the "check-move-result" signal asserted by SE4 when the move was selected. SE11 takes back the successful move, evoking UB2 to restore the board, and signals that the move at the previous depth is refuted. SE10 responds to the "refuted" signal, using the "check-move-result" that was asserted when E6-E7 (advancing the pawn) was selected by SE4. Generally, after a move is retracted by SE10 via UB2, other move candidates are tried (SE4), other strategies at the same level are tried (imagine a "select-strategy-move" re-asserted by the super-action of SE10), or SE2 and SE3 take effect. In our example, White abandons the attempt to advance the pawn, its level is decreased to 4, and the search continues in a similar fashion.

We now touch on a few points about the VAPs in Figure B.2 that were not brought out by the above. The treatment of terminal patterns recognized by instances of SE6 is according to one of the two procedures represented by SE8 and SE9. Recall that "check-terminal-position" is examined immediately when a new position is created in the search, so that the "succeed" or "refuted" signal to the previous strategy will occur before any other strategies are attempted at that new position. The terminal positions recognized by SE6 are general, as opposed to successes of particular strategies as represented by WBS2 - the result in either case is similar, though. A different kind of terminal position leading to the "terminal-win" signal in some cases is the maximum depth condition. Presently the maximum depth is 9, and when the search is 9 plies deep, SE7 fires (if the position is not terminal in any other sense), asserting "static-eval-strategy". If board conditions are right, in a rather optimistic evaluation, an instance of WBS3 fires; otherwise nothing further is done and the strategy at the previous depth succeeds for lack of refutation. The maximum depth cutoff is intended to be used only rarely, since the domain is rich in specific knowledge, so the present mechanism is only a stopgap, even though it is successful in the experiments described below. Finally, the MMC1 VAP represents a set of Ps that are evoked by WBS's to generate moves of desired classes, for instance moving toward a square. These means to generating move candidates are used whenever the desired move candidates cannot be easily constructed directly by the WBS's.

B.3. Full detail on selected aspects of KPKEG

As we have seen in the preceding subsection, KPKEG's six groups of Ps form the following functional units: the strategy executive, the board-updating operations, the means to strategies, the strategies themselves (two groups), and initialization. This subsection will indicate subdivisions of each of these groups, except the last. In most cases, typical Ps will be given to illustrate how certain kinds of chess knowledge are represented. Descriptions of all of the chess knowledge in KPKEG will be included. For the S Ps, we will give more detailed, abstract Ps which bring out issues of control. The listing for the actual program is in Appendix A, and a cross-reference is in Appendix B. Section B.4 is essential for decoding the actual Ps.

There are 55 S Ps, subdivided functionally into 9 groups as follows:

S0-S1: *initialization; 2 Ps.* [SE1]●
S3-S4, S15-S18: evocation of strategies, change of strategy levels; 6 Ps. [SE2-SE3]
S5-S9: tree mechanics, ascending and descending in search tree; 8 Ps. [parts of SE4, SE10, SE11]
S21-S210: selection of a move from the set of candidates; 4 Ps. [SE4]
S11-S13, S23-S26N: checking the results of an attempted move; 7 Ps. [SE10-SE12]
S30's, PN's, PW's, PV's (created by S60's): checking for terminal positions; 13 or more Ps. [SE5-SE7]
S40's: controlling actions for terminal positions; 3 Ps. [SE8-SE9]
S50's: printing the board externally; 3 Ps. []
S60's: recording the winning (terminal) positions as Ps; 9 Ps. [SE13]

The basic control in the executive corresponds to the VAPs SE2-SE4, SE8-SE12, and the RHS of SE5 (i.e., the second through fifth and the seventh group of S Ps). Figure B.3 gives abstract Ps (APs) that elaborate on those VAPs. Each AP has the VAPs and actual Ps to which it corresponds. Using the APs, we can get a more detailed picture of the control flow. The process of finding a move starts when the initialization asserts "select-strategy & check-other-strategy" (the latter signal is synonymous with "check-strategy-result" in the VAPs). If a strategy produces move-candidates, S2a will select one by using first a "max" metric, which takes the distance between two squares to be the maximum of the

● Square brackets enclose the names of the corresponding VAPs from Figure B.2.

```
S0a: [SE2, SE10; S3] check-other-strategy & depth & not select-strategy & not succeed
          & not move-candidate's & not refuted
     -> select-strategy & check-other-strategy;
S0b: [SE3; S4] check-other-strategy & depth & select-strategy-unresponded-to
     -> change-level;
S0c: [SE4; S5-S6] descend(move) & depth & (current-level OR level-from-preceding-depth)
          & current-mover
     -> make-move & check-terminal-position & erase-check-terminal-position
          & increase-depth & establish-level-at-new-depth & mover-is-other-player;
S0d: [SE10, SE11; S7-S9] ascend(move) & depth & current-level & current-mover
     -> erase-strategy-tried's & retract-move & restore-captured-pieces
          & decrease-depth & mover-is-other-player
          & erase-strategy-signals-from-depth-being-ascended-from;

S1a: [SE11, SE12; S11-S13] succeed(move,depth)
     -> ascend(move) & refuted(previous depth) OR make-the-move-if-depth-1;
S1b: [SE2, SE3; S15-S18] change-level & depth & current-level
     -> select-strategy & check-other-strategy
          & current-level(decreased if depth = 1 OR increased if depth > 1)
               OR depth [in case all levels have been tried];

S2a: [SE4; S21] move-candidate & depth & not check-move-result
          & not move-offboard & not move-onto-piece-of-own-color
          & not move-candidate-whose-destination-square-is-closer-to-pawn's-queening-
               square-by-max-metric-or-same-by-max-metric-and-closer-by-min-metric
          & not move-candidate-equal-by-previous-test-and-with-destination-square-
               lexically-less-or-destination-same-and-origin-square-lexically-less
     -> descend & check-move-result;
S2b: [SE3, SE11; S23, S26-S27] check-move-result & not refuted & depth(one deeper)
          & not move-candidate-at-depth-one-deeper
          & not other-strategies-to-be-checked-at-depth-one-deeper
     -> erase-move-candidates & record-win & succeed;
S2c: [SE8, SE11; S24] succeed-strategy -> refuted(previous depth);
S2d: [SE10; S25] check-move-result(depth) & refuted & depth(one deeper)
     -> ascend(refuted move);

S3a: [SE5; S38] erase-check-terminal-position -> select-strategy & check-other-strategy;

S4a: [SE8; S41] terminal-win(for mover) & depth
     -> refuted(previous depth) & not erase-check-terminal-position;
S4b: [SE9; S42-S43] terminal-win(for opponent) & depth
     -> check-other-strategy & all-levels-have-been-tried;
```

Figure B.3 APs for control in the executive

absolute values of the differences between their corresponding numerical coordinates; for equals by the "max" metric, S2a applies a "min" metric, which is similar except that the minimum is taken into account; when there are still contending candidates after those tests, lexical order is used.

S0c then carries out the bookkeeping involved in descending a ply, and evokes the Q Ps via "make-move" to update the board for the move selected by S2a. In descending, the usual action is that the mover at the new ply inherits the strategy level from the preceding move that it makes in the current search variation, that is, from two plies back. The level from one ply back is used in going from depth 1 to depth 2. This inheritance of levels injects some continuity into the search, since a player first tries to continue what he was trying on his preceding move. After the board is updated for the selected move, control returns from the Q's to examine the "check-terminal-position" signal asserted by S0c. Terminal positions are recognized by a set of Ps not shown (discussed below), and if nothing is recognized, S3a fires and the strategy selection is started at the new depth, as before.

There are three ways for the descent in the search tree to stop: the recognition of a terminal position, the recognition of the success of a strategy, and the exhaustion of all possibilities, which is a failure of a strategy. Terminal positions (including maximum search depth, which is terminal in a weak sense only) are checked in response to the "check-terminal-position" signal, asserted only when a new position is first entered from a lesser depth (closer to the root of the tree) - not when a position is re-instated from a greater depth (descendent node). If a terminal position or explicit success occurs, "terminal-win" is asserted and processed by S4a and S4b. S4a specifically refutes the strategy at the previous depth; the "refuted" signal is processed by S2d. S4b sets up an exhaustion condition so that S2b will get control, resulting in a success at the previous depth. S2b recognizes a failure of one strategy, implying the success of another, by noting that a move has not been refuted by the strategy at the descendent node (that strategy has tried all its possibilities with no success). The implied success is signalled by "succeed", which is picked up by S1a.

S0d carries out the bookkeeping of the actual ascent to the parent node in the search tree, evoking Q's with "retract-move" to update the board. After the ascent, control falls back to one of two places: to S2d if "refuted" is present (from S21a), which continues to propagate results back one more ply; or to S2a or S0a if there was a success at the descendant node that refuted the move made at the present depth (S2d). S2a selects from any move-candidates that are still available, but if none are there, S0a fires and strategy selection is evoked again.

Strategy selection is driven by S0a and S0b. S0a evokes a strategy (to generate move-candidates) via "select-strategy" and at the same time asserts a signal to which S0a or S0b respond. A strategy consumes the "select-strategy" signal and also asserts a "strategy-tried" signal (not shown except that S0d erases all such during ascent) so that no duplication can occur. Some strategies do respond with move-candidates in several sets, iterating through S0a, but when no further response is possible, S0b fires and the strategy level is changed, via S1b. Levels are changed in two ways depending on depth: at depth 1, which is the depth of the player trying to make an actual external move, the level starts out at the maximum (highest aspiration) and decreases when things don't work; at other depths, the level starts out at the level inherited from the ancestral (parent) node as explained above (S0c) and increases up to the maximum (in accord with the strategy hierarchy principle). When the maximum is reached, the action represented by the second half of S1b's RHS (after the "OR") is taken, and the "depth" signal is picked up by S2b. When the minimum is reached (depth 1 only), the program has failed to find a move to make.

To summarize the discussion so far, there are two aspects of the strategy executive: chess and PS control. There are several points with regard to the former. The executive does a fairly standard tree search, but it uses success or failure of strategies to evaluate positions rather than a more conventional material criterion. Strategy levels are inherited from parent or ancestral nodes, so that some unity of play over various depths in the tree is evident. Strategy levels start high and decrease at depth one, and start at the inherited value and increase at the other depths. Recognition of terminal positions occurs when a position is examined for the first time.

PS control is primarily of a fall-back nature: a move is made, for instance, and a Working Memory instance records it; when it has been processed, control falls back to examining that record and proceeding accordingly. In addition to checking move results, this occurs when strategy levels are exhausted ("depth" from S1b), during ascent ("depth" from S0d - note that S0a and S2a have explicit exclusion conditions to determine the appropriate action), when strategies are tried ("check-other-strategy"), and when terminal positions are recognized ("erase-check-terminal-position", S3a). Another kind of control is used for generating move candidates and for selecting strategies: When a strategy fires, it asserts a signal that inhibits future firings in the same context. Move-candidates exist as a set in Working Memory, so that when S2a is examined, a new one from the set is found (and erased). When there are no more candidates, control falls back to S0a and S0b.

Continuing now with more details of KPKEG, the Ps corresponding to VAP SE6 (Figure B.2), the S30's, encode conditions for recognizing ten terminal positions as follows:

a. A pawn on the eighth rank that cannot be captured by the enemy king; conditions i. and j. below are excluded; this is defined to be a win for White. (S31)
b. No pawn on the board (it has been captured); this is a draw (which is considered a failure for White). (S32)
c. The black king stalemated. (S33)
d. Checkmate. (S34)
e. The black king with the opposition and the white king not directly in front of the pawn; condition h. is an exception. (S35)
f. The white king on the same file as the pawn, two or more squares in front of it, and the black king not closer to the pawn than the white king. (S36)
g. The white king on the square in front of the pawn, with the opposition (to be defined below). (S36O)
h. The white king on the sixth rank, in front of the pawn somewhere and fairly close, and the black king not closer to the pawn. (S36R)
i. A special stalemate condition with the pawn just promoted at C8, black king at A7, and white king controlling B6. (S37L)
j. Similar to i., but reflected to the right side of the board (F8, H7, G6). (S37R)

These may not be correct or powerful enough from the chess standpoint (see Fine, 1941) but they suffice for present first-approximation purposes. "Opposition" is an endgame term that is defined narrowly as a situation in which the kings are on the same file with one intervening square; the player not on the move has the opposition. This set of Ps is augmented by specific patterns added as Ps, which recognize specific board situations that have been determined during search to have a known eventual result.

An example of how one of these conditions is expressed is in Figure B.4. Refer to Section B.4 for predicate meanings.

```
  S36; "WK FRONT2+" :: CHECK:TERM(D,P) & NOT SATISFIES(D,D EQ 1) & KPK:HASP(C1)
bind the kings and the pawn to variables:
        & ISKING(A1) & HASCOLOR(A1,C1) & ISKING(A2) & VNEQ(A2,A1) & ISPAWN(A3)
establish rank and file for locations of white king and pawn; both on same file:
        & LOC(A3,S1) & RF(S1,R1,F1) & LOC(A1,S2) & RF(S2,R2,F1)
white king two or more in front of pawn:
        & SATISFIES2(R1,R2,R2 ?+GREAT R1+1)
location, rank, and file of black king:
        & LOC(A2,S3) & RF(S3,R3,F3)
black king not closer to the pawn than the white king:
        & NOT SATISFIES3(R2,R3,F3,MAX(ABS(R3-R1),ABS(F3-F1)) ?+LESS R2-R1)
    => TERM:WIN(C1,'S36) & NEGATE(1) & NOT ERS:CHECK:TERM(D,P);
```

Figure B.4 Implementation of terminal position f.

The Ps corresponding to the UB (updating board) VAPs (Figure B.2), the Q's, are grouped as follows:

Q0-Q0c: print a move trace; 2 Ps.
Q1-Q2: initiate move retractions; 2 Ps.
Q3-Q4: move pawn forward and backward; 2 Ps.
Q7: detect illegal king move, i.e., into check; 1 P.
Q8-Q9: bookkeepping for any capture move; 3 Ps.
Q10-Q19: king moves; 9 Ps.

A move is given as an origin square and a destination square. The Q's print a trace, detect the type of piece to be moved, determine the direction of the move, change the location of the piece, detect captures, save information about a captured piece so that it can be restored, and update the squares controlled by a piece as it moves.

Of the two types of moves in KPKEG, the king move is more typical of the majority of chess moves than is the pawn move. The actual move is done in two steps (P firings), one to do the part common to all directions for the move and the other (one of 8 Ps) to do direction-specific updating. The split into two is largely for reasons of economy of expression. All eight directions are distinct because of the board representation, which uses a different predicate to show the relation of a square to each adjacent square. Figure B.5 gives the common P and one of the directional Ps.

The M Ps, corresponding to VAP MMC1, are divided into five groups:

M1-M8: generate move candidates to move toward a square; 8 Ps.
M9-M9N: special cases for moving toward; 3 Ps.
M11-M14: handle the delayed assertion of move-toward candidates; 5 Ps.
M16: generate candidate to move to a square; 1 P.
M17: special case for moving to; 1 P.

The means to move candidates for strategies are quite important to reducing the number of necessary strategy Ps, since for moving in eight directions, different sets of move candidates are appropriate. There are three move candidates in the set for moving toward one square from another: one to a square approximately in the same direction as the target, and two that are adjacent to the first. Figure B.6 gives a typical means P.

```
  Q11; "K COMMON" :: MAKE:MOVE(S1,S2) & LOC(A1,S1) & ISKING(A1) & NOT OFFBOARD(S2)
test that move isn't onto a controlled square:
      & NOT( EXISTS(C1,C2,A2) & CONTROLS(A2,S2) & HASCOLOR(A1,C1)
            & HASCOLOR(A2,C2) & VNEQ(C1,C2) & NOT RETRACTING(S1,S2) )
check that the move isn't onto a piece of the same color:
      & NOT( EXISTS(A2,C) & LOC(A2,S2) & HASCOLOR(A1,C) & HASCOLOR(A2,C) )
make sure that the square is reachable:
      & CONTROLS(A1,S2)
signal for capture check and direction-specific component:
   => CHECK:CAP(A1,S2) & MAKE:MOVE:K(A1,S1,S2)
and do the updating common to all king moves:
      & LOC(A1,S2) & CONTROLS(A1,S1) & NEGATE(1,2,7) & NOT RETRACTING(S1,S2);

move the king diagonally left-forward:
  Q16; "K DIAGLF" :: MAKE:MOVE:K(A1,S1,S2) & DIAGLF(S1,S2)
establish the squares whose control will change:
      & DIAGRF(S1,S3) & CONTROLS(A1,S3) & DIAGRB(S1,S4)
      & CONTROLS(A1,S4) & RANKR(S1,S5) & CONTROLS(A1,S5) & FILEB(S1,S6)
      & CONTROLS(A1,S6) & DIAGLB(S1,S7) & CONTROLS(A1,S7)
      & DIAGRF(S2,S8) & DIAGLF(S2,S9) & DIAGLB(S2,S10) & FILEF(S2,S11)
      & RANKL(S2,S12)
make the changes (2 controlled squares stay controlled):
   => CONTROLS(A1,S8) & CONTROLS(A1,S9) & CONTROLS(A1,S10) & CONTROLS(A1,S11)
      & CONTROLS(A1,S12) & NEGATE(1,4,6,8,10,12);
```

Figure B.5 Updating the board for a king move

```
  M1; "MOVE TW DRB" :: MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
determine that the direction is diagonally right-backward, using rank and file coordinates:
      & RF(S1,R1,F1) & RF(S2,R2,F2) & SATISFIES2(F1,F2,F1 ?-LESS F2) & SATISFIES2(R1,R2,R1 ?-GREAT R2)
locate the appropriate three squares and set up the moves:
      & RANKR(S1,S3) & FILEB(S1,S4) & DIAGRB(S1,S5)
   => MOVE:HOLD(D,S1,S3) & MOVE:HOLD(D,S1,S4) & MOVE:HOLD(D,S1,S5) & NEGATE(1);
```

Figure B.6 Means for moving toward a square

The bulk of the chess knowledge in KPKEG is in the strategy Ps, the W's and B's, corresponding to the WBS VAPs in Figure B.2. As indicated in the VAPs, the knowledge is represented three ways: one for move-candidate generation, one for recognizing immediate success, and one for making a maximum-depth static evaluation. Since the three are somewhat similar, we consider details for the first only, in Figure B.7. Again, as for the terminal-position chess knowledge, no claim is made for correctness of these strategies in general. But they are adequate as a first approximation, and from the present limited success, we conclude that PSs are adequate for encoding whatever the correct knowledge is. The relation between the last two columns in Figure B.7 is that at the same level the strategies are (intended to be) opposites: success of one refutes the other. The levels are (intended to be) such that success of a strategy at a higher level refutes a move from a lower level, but not vice versa. A typical strategy P is given in Figure B.8.

Level	P group	White	Black
7	B1	Checkmate (impossible in K-P-K)	Capture pawn
6	W2, B2	Queen the pawn, move to 8th rank	Stalemate
5	W3, B3	Advance pawn, move king off square in front of pawn	Intercept pawn by moving toward pawn's queening square
4	W4, B4	Control path of pawn by moving king toward the square two in front of the pawn	Block pawn by moving toward any square in the pawn's path
3	W5, B5	Defend the pawn by moving toward it	Attack the pawn by moving toward it
2	W6, B6	Move toward the enemy king, to restrict its movement; always fails at depth 2; try to gain the opposition	Same as for White
1	both W7	Any move not toward the enemy king and not toward the pawn; always fails at depth 2	Same as for White

Figure B.7 Summary of chess knowledge in the strategy Ps

```
  W4; "CONTR P" :: SELECT:STRAT(D,P) & KPK:HASP(P) & CUR:LEVEL(D,L) & SATISFIES(L,L EQ 4)
       & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W4) )
bind pawn and white king:
       & ISPAWN(A1) & ISKING(A2) & HASCOLOR(A1,C) & HASCOLOR(A2,C)
find the square two in front of the pawn:
       & LOC(A1,S1) & FILEF(S1,S2) & FILEF(S2,S3) & NOT CONTROLS(A2,S3)
evoke means and indicate the strategy has been tried:
    => MOVE:TOWARD(D,A2,S3) & STRAT:TRIED('W4,L,D) & NEGATE(1);
```

Figure B.8 A typical strategy P

B.4. Meanings for KPKEG predicates

Two sets of KPKEG predicates are central to the program and to the representation of the game, and are given here to provide an index into the following alphabetical list:

Search: ASCEND, CHANGE:LEVEL, CHECK:MOVE:RESULT, CHECK:TERM, CHECK:OTHER:STRAT, CUR:LEVEL, DEPTH, DESCEND, MAKE:MOVE, MOVE:CAND, REFUTED, RETRACT:MOVE, SELECT:STRAT, SUCCEED.

Board representation: CONTROLS, DIAGLB, DIAGLF, DIAGRB, DIAGRF, FILEB, FILEF, LOC, OFFBOARD, RANKL, RANKR, RF.

The following are the types for the arguments of predicates in the description below:

a	actor, i.e., particular piece	l	level (of strategy)
c	color	p	player
d	depth	r	rank
f	file	s	square

ASCEND(s1,s2)	ascend to a lower ply by retracting the move from s1 to s2. (S)●
CAPTURED(a,s,d)	at d, a was captured and removed from s. (S, Q)
CHANGE:LEVEL(d)	change the strategy level at d. (S)
CHECK:CAP(a,s)	check if there are any captures by a moving onto s. (Q)
CHECK:MOVE:RESULT(d,s1,s2)	check the result of the move made from s1 to s2 at d. (S)
CHECK:OTHER:STRAT(d,p)	check for other strategies for p at d, after at least one strategy has been tried. (S)
CHECK:TERM(d,p)	check if the current position (at d) is a terminal one; p2 is to move. (S, PN)●●
CONTROLLED(a,d,s)	a controlled s (see CONTROLS) before it was captured in the search at d. (S, Q)
CONTROLS(a,s)	a controls s, in the sense that it can move directly onto s. (all but PN)
CONTROLS:K(a)	set up the CONTROLS instances for king a. (X)
CONTROLS:P(a)	set up the CONTROLS instances for pawn a. (X)
CUR:LEVEL(d,l)	l is the current strategy level at d. (S, Q, W, B, PN)
DEPTH(d)	d is the current search depth. (S, Q)
DESCEND(s1,s2)	move one ply deeper by moving s1 to s2. (S)
DIAGLB(s1,s2)	s2 is diagonally left and back from s1. (Q, M, W, X)
DIAGLF(s1,s2)	s2 is diagonally left and forward from s1. (Q, M, X)
DIAGRB(s1,s2)	s2 is diagonally right and back from s1. (Q, M, W, X)
DIAGRF(s1,s2)	s2 is diagonally right and forward from s1. (Q, M, X)
ERS:CHECK:TERM(d,p)	erase the corresponding CHECK:TERM; this signals completion of the check. (S, PN)
ERS:MOVES(d)	erase unexamined MOVE:CAND's at d. (S)
ERS:STRAT:TRIED(d)	erase STRAT:TRIED's at d. (S)
FILEB(s1,s2)	s2 is directly back along the file of s1. (all but PN)
FILEF(s1,s2)	s2 is directly forward along the file of s1. (all but PN)
FINDMOVE(p)	find a move for p; typed by user. (S)
HASCOLOR(a,c)	a has color c (B or W). (all but M)
ISKING(a)	a is a king. (all but M)
ISPAWN(a)	a is a pawn. (all but M)
KPK:HASP(p)	this is a K-P-K game; p has the pawn. (S, W, B, PN)
KPKINIT(x)	initialize for a K-P-K game; x is a dummy. (S, X)
LAST:PN(x)	production x is the last one added to the position-net module. (S)
LOC(a,s)	a is located on s. (all)
MAKE:MOVE(s1,s2)	make the move from s1 to s2. (Q)
MAKE:MOVE:K(a,s1,s2)	update the board (CONTROLS) for the king move of a from s1 to s2. (Q)
MAKE:MOVE:T(s1,s2)	print the trace message for the move from s1 to s2, then signal MAKE:MOVE. (Q, S)
MAXDEPTH(d)	d is the maximum depth for the search. (S)
MAXSLEVEL(p,l)	l is the maximum strategy level for p. (S)
MEANS:EXAM(d)	signal that MOVE:CAND's are not to be generated by a means (M Ps) at d, but rather the potential moves are to be held for examination (MOVE:EXAM). (M, W)
MEANS:HOLD(d)	hold the emission of MOVE:CAND's at d from a means (M Ps) until all possibilities are ready. (M, W, B)
MEANS:RELS(d)	release the moves held back by MEANS:HOLD at d. (M)
MINSLEVEL(p,l)	l is the minimum strategy level for p. (S)
MOVE:CAND(d,s1,s2)	the move from s1 to s2 is a candidate at d. (S, M, W, B)

● The initials appearing at this place refer to P groups to which a predicate is relevant.

●● PN stands for Ps in the position net generated by the RECORD:BLD process.

MOVE:EXAM(d,s1,s2)	the move from s1 to s2 is ready for examination by a strategy (see MEANS:EXAM). (W, M)
MOVE:HIST(x)	x is a list of the moves made in descending to the current depth, used for external display only. (S)
MOVE:HOLD(d,s1,s2)	s1 to s2 is a potential MOVE:CAND at d, generated by a means. (M)
MOVE:TO(d,g,s)	generate moves to get g to s at d. (M, W, B)
MOVE:TOWARD(d,g,s)	generate moves toward s from g's present location. (M, W, B)
MOVER(p)	p is the color to move in the current position. (S)
MOVING(p,g,s1,s2)	for external display, p is moving g from s1 to s2 as a real game move. (S)
NODE:COUNT(x)	x is a count of the number of nodes searched, for external display. (Q, S)
OFFBOARD(s)	s is off the board; it exists as a dummy location to simplify the board patterns. (S, Q, M)
PLAYER(p)	p is a player, either B or W. (S)
PRINT:BOARD(x)	print the board externally; x is a dummy. (S)
PRINTED:BOARD(x)	the board has been printed; x is a dummy; this is used to prevent the board display twice with no intervening changes. (S, Q)
RANKL(s1,s2)	s2 is directly to the left of s1, same rank. (Q, M, W, X)
RANKR(s1,s2)	s2 is directly to the right of s1, same rank. (Q, M, W, X)
RECORD:BLD(d,l,s1,s2,x)	ready to add a set of Ps to the position net of terminal positions, which recognize that s1 to s2 is the key move; d and l are the depth and level at which the importance of the position were determined and x is a list that is the common part of the LHSs of the set of Ps. (S)
RECORD:DONE(d,x)	the P whose tag (PN, PV, or PW) is x has been recorded at d in the RECORD:BLD process; this prevents duplication. (S)
RECORD:FIN(d)	the main part of the RECORD:BLD process is finished at d. (S)
RECORD:FIN2(d)	finish the RECORD:BLD process by erasing various intermediate data. (S)
RECORD:PRE(d,l,s1,s2,s3,s4,s5,p)	at d and l, s1 to s2 is the key move leading to a terminal position (see RECORD:BLD); p is to move, and s3, s4, and s5 give the positions of the pawn, white king, and black king. (S)
RECORD:WIN(d,s1,s2)	record the terminal position, see RECORD:BLD, at d, key move s1 to s2. (S)
REFUTED(d)	the strategy at d is refuted, at least with respect to a particular move (CHECK:MOVE:RESULT). (S, Q)
RESTORE:CAP(d)	restore the captured piece removed by a capture move (CAPTURED), at d. (S)
RESTORE:CON(g,d)	restore the CONTROLS removed by a capture move (CONTROLLED); g was captured at d. (S)
RETRACT:HOLD(s1,s2)	hold the retraction (RETRACT:MOVE) of the move s1 to s2, since it was never made due to illegality. (Q)
RETRACT:MOVE(s1,s2)	retract the move from s1 to s2, restoring the board state to its previous condition; the reverse of MAKE:MOVE. (Q, S)
RETRACTING(s1,s2)	s1 to s2 is being retracted; this suppresses certain legality checks for king moves. (Q)
RF(s,r,f)	s has rank r and file f, both numbers. (S, M, W, B)
SAVE:CON(g,d)	save the CONTROLS of g, at d, as CONTROLLED. (Q)
SELECT:STATIC(d,p)	at d, do a static strategy estimation for p. (W, B, S)
SELECT:STRAT(d,p)	at d, do a (dynamic) strategy selection, which generates move candidates, for p. (S, W, B)
STATIC:EVAL(d,p)	signal that STATIC:EVAL is appropriate in the current position; this affects the direction of processing after CHECK:TERM. (S)
STRAT:TRIED(x,l,d)	strategy x (the name of a P) has been tried at d and l. (S, W, B)
SUCC:STRAT(d,p,l,x)	strategy x (the name of a P) has succeeded for p at d and l; the success is known statically, without further search. (S, W, B)
SUCCEED(d,s1,s2)	at d, the move s1 to s2 has succeeded, in the strategic sense. (S)
TERM:WIN(p,x)	p has a terminal win (for White, a chess win, for Black, a draw), indicated by P x; at maximum depth this evaluation is static and not as strictly a win (see SELECT:STATIC). (S, W, B, PN)
TESTn(x)	initiate the test problem n, n = 1, 2, 3; typed externally by the user. (X)
TRACING(x)	a dummy predicate used to show the printing of an external trace. (S, Q)

W6W:RES:EXAM(d,g)	examine the results of the means evoked by P W6W (moving g at d) using MEANS:EXAM; W6W is a static estimator (SELECT:STATIC) that uses simply the existence of one of a class of moves. (W)
W7:RES:ERS(d)	erase the results of the W7:RES:EXAM process. (W)
W7:RES:EXAM(d,g)	at d, examine the results of the means evoked by P W7 using MEANS:EXAM; W7 desires moves that are not generated by the means. (W)
W7W:RES:EXAM(d)	similar to W6W:RES:EXAM. (W)
WIN:CAND(d,s1,s2)	at d, s1 to s2 is a candidate that has led to a win in an identical situation at a different depth; see RECORD:BLD. (S, PN)

C. Results of Experiments with KPKEG

KPKEG has been tested on three simple problems, called Test1, Test2, and Test3. These are not intended to be representative of the class of all K-P-K positions, but KPKEG's behavior does demonstrate that it is an adequate basis for a more complete program. Test1 is discussed in detail in Section B, and is exhibited in Figure B.1. Appendix D examines KPKEG's behavior on Test1 in detail, exhibiting: the program trace, showing search behavior in the tree of chess moves; the state of Working Memory after the run, which includes the internal board representation; the trace of P firings corresponding to the program trace, broken into distinguishable corresponding segments; and a control flow summary trace, which breaks P firings into groups. Appendix E contains four more program traces, two for some experimental options on Test1, and one each for Test2 and Test3.

Test1 is a good test because it requires more searching than the typical K-P-K position. This searching exercises KPKEG's executive Ps and results in the evaluation of a variety of terminal positions. It also allows meaningful comparison of effects of various options on the search. KPKEG's behavior on Test1 has been described in some detail in Section B.1. The traces on Test1 in Appendix E are of primary interest here. Two search options explored with Test1: (1) The procedure of decrementing strategy levels from the maximum at depth 1, but passing down strategy levels to other depths, and incrementing from those to the maximum. (2) The storing of winning positions for future use in the search. The standard version of KPKEG, with both of these options turned on, finds a move for Test1 by searching 40 nodes. A version with the strategy level changed to decrements at all levels searches 80 nodes (the first trace in Appendix E), and a version without the position storing searches 60 nodes (the second trace in Appendix E). The combination with both options in their non-standard setting was not tried. This is good evidence, at least as far as a single test position can provide, that the standard version has the proper options.

The most significant change in KPKEG's behavior on Test1 results from an experiment not shown: if the carrying down of strategy levels from two plies back is not done (Ps S5 and S6 become S5 modified to work at all depths so that the level from one ply back is used), the search goes on for hundreds of nodes and fails to find a satisfactory move for White. One critical point is the situation at node 35 (please refer to Appendix D), where Black is at level 5 but White responds at level 4, as in the sequence leading to node 23 (which happens to be caught by the position net, PN-5); in the alternate version, White is forced to be at level 5, and only tries to advance the pawn, failing to refute Black's move and eventually failing at depth 1 with the move E4-E5. This demonstrates that the alternative is detrimental to the evaluative effectiveness of the program.

Test2 and Test3 (Figure C.1) are rather similar as starting positions, but have some interesting differences in their search. Their shared traits are more important than their differences. Both tests show the application of the kind of specific knowledge that is typically applied in K-P-K positions. In particular, White searches very few nodes, four or less, in finding a winning move. Black, on the other hand, searches many more in its futile attempts (it would probably be more reasonable for the program to resign in situations

```
8  ..  ..BK..  ..       8  ..  ..BK..  ..
7..  ..  ..  ..         7..  ..  ..  ..
6  ..  ..WK..  ..       6  ..  ..WK..  ..
5..  ..  WP  ..         5..  ..  ..  ..
4  ..  ..  ..  ..       4  ..  ..WP..  ..
3..  ..  ..  ..         3..  ..  ..  ..
2  ..  ..  ..  ..       2  ..  ..  ..  ..
1..  ..  ..  ..         1..  ..  ..  ..
  A B C D E F G H         A B C D E F G H
```

Figure C.1 Test2 and Test3 initial positions; White to move

whose value is known). For illustrative purposes, the search is useful because it exercises some more of the terminal-position recognizers, and makes use of strategy Ps (W's and B's) for the lower levels (Test1 only got as low as level 4).

D. Production-System-Related Features of the Implementation

KPKEG's organization makes programming by gradually adding Ps easy. There is a clear division into the strategy executive, the strategies, the means, the terminal patterns, and the board-updating process. KPKEG was built up by leaving strategies and terminal patterns unspecified until the executive was in good shape. The action of the unspecified parts was easily filled in by manual intervention at pre-arranged points. The executive developed from an initial approximation by adding Ps to represent new cases of necessary action and by modifying the existing Ps to be more discriminating. For instance, there are many ways that a move can be refuted or allowed to succeed (APs S1a, S1b, S2b, S2c,S2d, S4a, and S4b in Figure B.3), and these ways developed gradually as tests were tried. When the executive was in good shape, strategy Ps and terminal patterns were added, resulting in more executive modifications as still more was found out in doing tree search over a wider range of positions. The options for the executive discussed in Section C were not tried until all the gaps were filled in. Two features of this mode of programming are very dependent on using PSs: each P does a relatively small manipulation to a global Working Memory (half a dozen or fewer changes), and the action of the unspecified modules is usually the firing of just one P, even in their final form.

Several kinds of control are exhibited in KPKEG: iterating through sets of things to be tried, evoking some process and at the same time asserting a second signal to which control will fall back, and factoring a complex selection or decision process into a cascade of P firings. The executive iterates over strategies by repeatedly evoking the strategy Ps to get move candidates. The strategy Ps each assert a Working Memory item that prevents repetitions at the same depth, amounting to a simple way of keeping the context of the generation. Within each strategy the order in which Ps fire is indeterminate, but there could easily be more control, with nothing added to the executive. Another form of iteration through a generated set is in using the move candidates asserted by a single strategy. Each time control falls back to P S21 (AP S2a), it selects (with one firing) one of the candidates, and erases it so it won't be considered again.

The RHS of AP S0c illustrates the way control can be arranged to fall back to process signals stacked up in Psnlst's :SMPX. First, it evokes the Q's with a "make-move" signal, and control falls back to the stacked "check-terminal-position" signal (the second conjunct in S0c's RHS). This results in evoking the terminal position patterns (S30's) and if none fires, control flows to a P that responds to "erase-check-terminal-position", also asserted by S0c (see AP S3a). When an exhaustion of strategies occurs (all levels tried at some depth), control falls back to the appropriate place by re-asserting the DEPTH instance (AP S1b). The "new" DEPTH is then examined in connection with instances at the previous depth that recorded what was being tried when the descent occurred, in order to check the results. A new DEPTH is also responded to when an ascent occurs (a more specific signal is not used), and the response varies according to whether move candidates and untried strategies exist (APs S0d, S0a, S0b, S2a) - here the response is selected from a range of possibilites, illustrating the potential for openness of control.

Control through the factoring out of cases is evident in two places as a result of the board representation, which distinguishes eight inter-square relations. The king-move Ps

(Q10's) consist of one P that fires for all king moves plus a set of eight, one of which fires to finish the move. The means Ps for moving toward some square are also eight in number. A strategy P decides what it is to move toward and a means P fires to produce that actual move candidates. This cascading of selections from among sets of Ps is the essence of PS control: action sequences alternate with complex selections of what is to occur next, which allow potentially the application of large amounts of knowledge. As more knowledge is applied in directing control flow, more intelligence will result in the overall process.

Most of the <u>chess knowledge</u> in KPKEG is encoded in the S30's, the W's, and the B's, whose content was discussed in Section B.3. The knowledge is exclusively in the form of patterns for recognition (LHSs), with relatively simple actions (RHSs). The patterns consist of testing: locations and controlled squares for the chess pieces, inter-square relations, numerical rank and file properties, inter-piece distance, and relations of pieces with each other and with the edges of the board. The actions are "terminal-win" signals, move candidates, or signals to evoke means to move candidates. This simplicity is due to the simplicity of chess knowledge (at least in K-P-K), the condition-action nature of PSs, and the organization of KPKEG into executive, strategies, etc. Note that even though the Ps representing chess knowledge are not independent of the containing strategic control, and thus include control signals, the control is minimal and uniform over functionally similar Ps. The design philosophy is to establish a flexible matrix into which specialized knowledge is added. It is not necessary to limit added knowledge to single-P packages, as is illustrated in several places (e.g. the W7 Ps). The general properties of KPKEG allow easy encoding of chess knowledge, but the syntactic features could stand improvement, as we will discuss in the next paragraph.

Several features of the PS architecture are especially <u>awkward or inefficient</u> for the chess task in particular. (1) The primary inefficiency in KPKEG is in finding one match among a set of Ps that are constructed such that only one match (or perhaps a small number) in a given situation is likely. This is the case for most of the chess knowledge, i.e., the strategies and the terminal-position patterns. The opportunity for savings is that failing one match from the set might be used to reject some set of Ps from consideration. A simple and effective remedy is to store (and perhaps represent externally) the Ps as a tree of tests, where rejecting some branch in the tree amounts to rejecting the set of Ps whose RHSs correspond to that subtree's terminal nodes. (2) A related problem is a certain repetitiveness of bindings in the patterns. For instance, many of the patterns start out by binding variables to the locations of the kings and the pawn. This problem can be remedied in the same way as the preceding one. (3) The Working Memory for the board representation predicates is heavily loaded, probably resulting in high costs for patterns that access a number of board relations. Since, at present, the instances of each predicate are implemented simply as a list, there is room for improvement. The match routine could be modified to evoke functions to compute relations, perhaps resulting in a significant cost saving over the present access of a long list. (4) There are probably a number of recurring pattern expressions of a chess-specific nature that could be made more easily expressible by syntactic conventions. These could be obtained by detailed study of existing Ps and by analysis of chess knowledge. Further detail on this is beyond the present scope, since it appears applicable only to chess tasks.

E. A Comparison to a Similar Program in Lisp

KPKEG can be compared in detail to a similar program in Lisp, developed by Perdue (1975). Perdue's program, CP, can presently do tasks similar to KPKEG's, but is intended to develop into a much broader class of chess endgames. This section will first compare the overall organization in the two programs. Differences in chess knowledge content and in approach to the problem give rise to behavior differences, to be discussed second. Considering superficial aspects, such as conciseness and efficiency, also gives rise to contrasts, discussed third. Differences in the details of representations and processing will be discussed last.

The control organizations of KPKEG and CP are quite similar, ignoring for the moment that the means for implementing control are radically different. The main function in CP is Findamv (find-a-move), which controls the tree search, and calls other procedures to recognize terminal positions, to try making moves, and to do tree bookkeeping. Findamv is an iterative (as opposed to recursive) alpha-beta minimax procedure, looping over a body of code that either descends or ascends in the tree according to results of subordinate function calls. This corresponds roughly to the control parts of the S P group (i.e., excluding the S30's), which in effect loop by re-examining the "check-other-strategies" signal. The tree-bookkeeping functions correspond to S5-S7, and the functions called by Findamv to recognize terminal patterns correspond to the S30's. The major action of Findamv is to call the function Tryamv (try-a-move), which results in a new board position. Tryamv calls several functions in turn, the most important of which are More!Moves and Move2. Move2 actually executes chess moves and corresponds to the Q Ps. More!Moves has a producer-consumer relationship to the strategy function RG (recognize), and calls Genmvs (generate-moves) with the results of RG. It is "producer-consumer" because More!Moves calls RG repeatedly, each time obtaining something new, in much the same way as the S Ps repeatedly evoke the W's and B's. RG and its subordinate functions examine the board and propose strategies in correspondence with KPKEG's W and B Ps, except that RG produces an instantiated strategy descriptor rather than actual move candidates. More!Moves takes RG's output and passes it to Genmvs, which executes (Evals) the instantiated strategy descriptor to produce actual move candidates. Genmvs thus corresponds to the move-candidate assertion by the W and B Ps, and also to the M Ps.

In summary, the overall form of control organization is quite similar in the two programs. KPKEG maintains its control with explicit Working Memory items and by responding to new items in Working Memory, whereas CP uses the conventional Lisp control stack. But Ps in KPKEG group naturally into sets that functionally correspond to Lisp functions in CP.●

KPKEG and CP differ markedly in behavior, even though the control organization can be put into the above correspondence. CP is not strongly based on the strategy hierarchy principle, but rather does a mini-max alpha-beta search using more conventional evaluation procedures. Because of this and because of differences in the chess knowledge (e.g. the

● As far as I know, the organization of the two programs was developed independently.

patterns tested in CP's RG don't correspond exactly to KPKEG's W Ps), CP's search is shorter, covering around 10 nodes on KPKEG's Test1 as opposed to 40. CP is designed so that strategies tend to generate very few moves at each node, whereas KPKEG aims to make the strategies generate all conceivable moves that might lead to the strategic objectives at the particular strategy levels. In addition, CP doesn't search through alternatives when backing up, but returns all the way to the initial starting position and try new move sequences from there. Even though these differences give rise to different behavior, I maintain that they are non-essential, in the sense that they could easily be brought into line without changing the characteristics of the two programs on which the following comparisons are based.

There are a number of differences between KPKEG and CP that are primarily attributable to differences between Psnlst and Lisp, and secondarily perhaps to the difference in programmers. KPKEG has 140 Ps, with a listing of about 900 lines, whereas CP has about 270 functions with a listing of about 2640 lines. By these (very crude) measures, KPKEG is much more concise, a factor of 2 in elementary program units and a factor of 3 in size of program listing. In run-time efficiency, KPKEG is somewhat worse than CP, using 20 seconds per node (which turns out to be 20 P firings) as opposed to about 6 seconds. Section D contains a discussion of some possible causes for inefficiency in the PS, and suggests some modifications. In addition, it should be pointed out that the present PS is done by interpretation, rather than by compiling the Ps into some kind of optimal network, which would have the potential of speeding up the recognize-act cycle by avoiding duplication in condition testing (see Chapter VII).

The most marked contrasts between KPKEG and CP are in the relatively low-level details of how things are represented and processed. Where KPKEG uses Working Memory relations to represent the chess board, CP uses a two-dimensional array, accessing squares by their coordinates. The KPKEG representation is actually dual: one way expresses the eight intersquare relations (e.g., C3 to D2 is the DIAGRB direction), and the second way associates coordinates to the square names (e.g., RF(F4, 4, 6)). The dual representation is in part forced by a peculiarity of Psnlst, which doesn't allow constants to be expressed directly in the LHS match; using the coordinates as constants indirectly would force a search through 64 pairs of variable bindings. This becomes intolerable when one is testing for two squares' having some relation between them, requiring a search through 64 X 64 binding pairs to find the right set satisfying, say, some arithmetic predicate. (Even without the peculiar limitation, convenience in programming and readability of Ps might recommend the dual representation.)

A related feature is KPKEG's use of Working Memory for CONTROLS relations, where CP recomputes them each time they're necessary. CONTROLS is used to indicate that a piece can move directly onto a square, and is involved in testing, e.g., whether the pawn is safe on some square. For the king, for instance, CP tests control of a square by testing whether the king is on one of the eight adjacent squares, and that in turn is tested by simple arithmetic on the square's co-ordinates. To do this test by co-ordinates in KPKEG would not be combinatorial as mentioned above, but would be cumbersome, requiring testing of eight numerical predicates between the king's coordinates and the square's. In Lisp the cumbersomeness can be packaged into one function, but to do this "subroutining" in PSs would force breaking a single match into three, one to set up the test, one to do the test (one of eight Ps might fire), and one to finish matching the condition that included the test. Some clumsiness is still inherent in the PS implementation of CONTROLS, as is illustrated by the king-move Q Ps. There, eight Ps are required to do the CONTROLS updating when a king move is made, one P for each potential king-move direction.● Note that these eight are coded once, for each chess piece, so that there need be no concern along these lines in dynamic augmentation situations. But the use of extensive Working Memory relations like CONTROLS (as opposed to intensive recomputed relations) is a mechanism that is essential when relations become more complex, as they certainly do in chess, and the mechanism is provided by PSs as an essential architectural feature.

Both programs represent the board as a global structure that is updated and downdated as moves in the search are made and retracted. CP records necessary contextual information for the board at each depth in a stack that is correspondingly pushed and popped, whereas KPKEG uses a depth argument that is attached to predicates that store essential information such as captured piece locations.

CP keeps its strategies and move candidates in a similar structure, a context list whose head (Car) is a list of untried ones and whose tail (Cdr) is the list of old, tried ones. KPKEG's Working Memory only stores, for move candidates, the untried ones, and for strategies, the ones that have been tried (STRAT:TRIED). Each strategy P includes a condition to ensure that no STRAT:TRIED exists for it, to avoid duplication, whereas move-candidates are simply erased on use (this doesn't guarantee that different strategies or different Ps of the same strategy don't generate the same moves, which are then tried). For each entry in CP's board-context stack, there is that pair of lists, where KPKEG marks the elements with a depth argument. The way CP handles generation of candidates for these lists is to generate a full list and then test whether the elements of that list are on the appropriate context list. Under this regime, for instance, in the producer-consumer iteration between More!Moves and RG, a list might be produced, only to discover that all its elements had already been added to the context list. In practice, for the sizes of lists encountered in CP, this is apparently not prohibitively costly.

Finally, we examine the parts of CP where PS-like patterns are tested. CP uses uniform database procedures constructed for storing properties, whereas KPKEG uses the existing Working Memory. CP has two functions, FORALL and EX (Exists), which perform iteration over lists and selections from lists according to specifiable Lisp COND's, operations that are included in the PS match. CP's patterns also make more use of function calls to test various conditions than do the Ps in KPKEG. In CP, all of the pattern testing is under strict control and is embedded in variable-binding contexts that establish the data for the patterns. This is less true of KPKEG, although sets of Ps are under control of explicit Working Memory items asserted at specific points in the control flow. Figure E.1 gives a pattern roughly comparable to Figure B.4, illustrating the function-calling style of the Lisp patterns.

In PSs, control of which matches are done is potentially more flexible and efficient: in KPKEG, selection is from an unordered set of P conditions, whereas a Lisp function containing a set of tests is executed in a fixed pre-determined order. The order of testing of P conditions could thus be rearranged dynamically as different Working Memory states

● Some subroutining in the PS is used, however, to handle what is common to the eight, for program conciseness.

```
(PROG (WP WK BK)
      (SETQ WK (WK (TOPBD)))    % TOPBD = current board, at top of stack %
      (SETQ BK (BK (TOPBD)))
      (SETQ WP (CAR (PAWNLIST 'WHITE)))
      (MAKE (STATVAL (TOPBD))   % STATVAL = static evaluation, the end result of Estim %
            (COND ( . . . )
                     . . .
                 ((AND (EQ (RANK WK) (+ (RANK WP) 2))  % WK in front of WP %
                          % RANK returns the rank value of the location of a piece %
                       (<= (ABS (- (FILE WK) (FILE WP))) 1)   % <= is less than or equal %
                          % FILE returns the file value of the location of a piece %
                       (NOT   % BK to move and not 1 away from WP %
                        (AND (BTM)     % BTM = predicate for Black to move %
                             (= (DIST (FILE BK) (RANK BK) (FILE WP) (RANK WP)) 1))))
                                        % DIST returns distance between two squares %
                  (SUREWIN WHITE))      % SUREWIN returns a triple of probabilities %
                  . . . )))
```

Figure E.1 Fragment of Estim function of CP

occur. It is conceivable to code a Lisp pattern matcher that has desirable efficiency properties as long as patterns to be matched are not allowed to become too arbitrary. Efficiency could also be maintained in more arbitrary patterns by including heuristic information in patterns, to guide the matcher. This would make adding patterns more difficult, however. The PS approach is to adopt specific and perhaps stringent conventions which allow a general *procedure* to compute an optimal matching strategy. This is not to say that such a procedure has been developed yet, but there is some indication that the problem is tractable.

F. Extending KPKEG

This section will consider the forseeable problems in extending KPKEG to a more complete chess program. First, we consider some topics having to do with the executive and with the strategy hierarchy principle. Then, we consider how KPKEG might be extended to more complex domains. These will require a number of extensions to KPKEG's representational capabilities, such as more complex inter-piece relations and descriptions of dynamic situations. In the following, the emphasis will not be on details of such extensions, but on their demands on the capabilities of PSs.

In the course of the preliminary experiments with KPKEG already described, several features of the strategy hierarchy principle and the executive have come to light. In a past try of Test1 in which KPKEG arbitrarily chose to try E4-D5 as its first move at level 4, KPKEG didn't see an opportunity to take the opposition and achieve its strategic objectives because its strategy level was too high, above the level for the opposition strategy. In general, it seems to be the case that two things are not quite right: the present ordering in the hierarchy may not be correct, requiring experiments with alternative orderings; and the whole level-oriented focus may be too narrow, requiring opening it up somehow to allow strategies to take over that look much closer to being successful, rather than sticking to a strategy that requires more search and whose success is not strongly indicated in the present situation. With respect to re-ordering the strategy hierarchy, it would be easy to change the appropriate Ps to different levels by substituting a different level constant. But attention must also be given to whether the principle is itself unattainable with the fine distinctions between levels at present. Perhaps fewer than seven levels is more apropriate for K-P-K, or perhaps no ordering is correct in all situations.

With respect to the narrowness of focus, perhaps the most promising approach would be to set up a few specialized patterns that would match and redirect the program's attention when the board is changed, before the ordinary strategies are evoked. For instance, it might be useful to recognize situations where king moves result in having the black king move out of the square so that the pawn is clear to advance; or situations where the pawn is left open to attack in the course of some other strategic maneuvers. A more radical change to KPKEG would be to reorganize the strategies to be much more bottom-up, analyzing the board in terms of what looks possible, rather than top-down as at present, setting up goals to try particular things in a predefined order. This would probably require much better descriptive capabilities as described below.

Finally, with respect to the strategy hierarchy, on the tests tried there appears to be no need for the standard alpha-beta minimax procedure; i.e., the search always stays in the region above "alpha", converging on the best available move from above. A proof or refutation of this property may emerge as the principle is exercised on chess tasks that aren't as limited as K-P-K.●

● None of the difference of 40 nodes searched versus 10 nodes for CP are due to alpha-beta considerations.

There are simple variations on the present task domain that introduce new complexities and that may force major changes in the basic descriptive elements that the Ps. work with. The tests used for KPKEG deal with relatively localized situations, as opposed to ones requiring many moves to bring the pieces together for the localized rules to apply. Such a situation is illustrated in Figure F.1.

```
  ..  ..  ..  ..
..BK..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  WP
  ..  ..  ..  ..
..  WK  ..  ..
```

Figure F.1 A non-localized K-P-K position

This class of situation requires at least the use of special strategies that generate fewer alternative move candidates, and candidates that are more specifically directed toward particular distant squares, than the present move-towards means. It also requires that the maximum search depth be increased (from its present setting of 9) or allowed to be changed as the situation demands. (Perhaps maximum depth is the wrong approach, but there will probably remain the idea that at some point the situation requires a static evaluation such as the one done now when the maximum depth is reached.)

Tasks with more than one pawn introduce considerable complexity. A typical situation is given in Figure F.2. Some salient features of such tasks are: the necessity for the white king, as well as the black, to broaden its strategy to stop enemy pawn advances; the necessity to divide the board into two or more sectors of activity that are to some extent independent of other such sectors; and the necessity to describe relations between such sectors, with particular attention to the ways in which individual pieces can perform functions in more than one sector.

```
  ..  ..  ..  ..
..  ....BP..BP
  ..  BK  ..  ..
..  ....WPWP
BPBP  ..  ..  ..
WP  ..  ..  ..
  WPWK..  ..  ..
..  ..  ..  ..
```

Figure F.2 A task presently beyond KPKEG's capabilities

As we have pointed out several times above, advances in KPKEG's power depend on enriching its board representation. Three levels of descriptive organization can be distinguished: <u>relations</u>, which are computed directly from the board, for instance, CONTROLS in KPKEG; <u>chunks</u>, which combine several relations, usually labelling commonly recurring or important combinations; and <u>board sectors</u>, which are the semi-independent units of analysis described above in connection with more complex endgame tasks. For KPKEG, which already has relations to a limited extent as Working Memory items, it is feasible to have relations, chunks, and even sector divisions computed when the board is updated, by Ps that recognize conditions that make or break the descriptive units. These Ps would not need to be specifically evoked, but would work in a bottom-up fashion (the considerations of efficiency discussed in Section D would apply here). Note that in already having some relations, and in the proposed updating capability, KPKEG is superior to CP, where additional ad hoc procedures and calling conventions would be required. CP and other similar program structures would probably find it difficult to direct their activity in a recognition-oriented bottom-up mode, since the structure lends itself so easily to the contrary top-down mode. It is envisioned that having better descriptive capabilites would prove advantageous in expressing strategy Ps and similar patterns, in changing KPKEG to be more bottom-up as just described, and in allowing patterns such as those constructed by KPKEG itself to recognize terminal position classes instead of specific positions.

Several specific features of KPKEG are troublesome with respect to more ambitious applications to chess. One is the problem of using the present Ps for a game in which Black has the pawn. The Ps do not mention Black or White, using a Working Memory instance (KPK:HASP) to determine which color the pawn is. But Ps that test board configurations rely heavily on the orientation of the board: "forward" is always towards White's eighth rank (Black's home row). A solution might be to transform the entire board representation so that it would be reversed with respect to the external game but would internally match the white-pawn assumption. Another feature of KPKEG is the repetitiveness of the search. The specific strategies may be at fault for generating duplicate moves; the strategy hierarchy, or its implementation as seven levels, may be at fault; or it may simply be necessary to implement a more general mechanism to prune duplicates. The general mechanism might consist of Ps that would record the results of specific moves in specific situations so that all future searches could take advantage of past effort. This, of course, has benefits beyond simply preventing duplicates. It also raises an issue that is pertinent even to the present, limited P-building scheme. That is, how can the number of Ps added be ultimately controlled, so that the set of Ps converges to a more-or-less stable size, or at least somehow avoids all possible board placements for each pattern? Perhaps the convergence will occur when more powerful descriptive devices are used, e.g., the chunks mentioned above. Using more abstract descriptors of the board in this way would result in Ps with greater generality, and in fewer distinct Ps overall. An alternative is a scheme of generalization that might collapse a set of existing Ps into one according to a general procedure. At present, only indications of the need for further research can be put forward.

Finally, we briefly consider some requirements for improved chess programs as put forward by Berliner (1973, 1975a). The basis for the improvements to be considered is the idea of a <u>causality facility</u>, whose purpose is to determine why a search fell short of aspirations. It must differentiate between failure for superficial reasons (a particular move, for instance) or for deep ones (inherent features of the situation). The first specific improvement comes from the idea of building a <u>refutation description</u> as a result of a search that failed strategically. The refutation description includes features of the position and of the search that the causality facility proposes as essential to the failure. It is used by move generators that try to counter those features, thus giving the program a way of restricting available move choices. For K-P-K, the implementation of this idea would result in searches with fewer branches than in KPKEG, but with the option of generating specific extra branches to meet specific demands. Since move-generators are Ps in KPKEG, the immediate approach to try would be to build specific Ps sensitive to elements of a refutation description in Working Memory. The second improvement comes from the idea of lemmas. Lemmas are the followup of a causality analysis, functioning to reject lines of play on the basis of a description of a difficulty that is known to be fatal to all such lines of play. The PS approach to this involves building a P to act as a "demon" to recognize such situations and immediately refute moves that don't surmount the difficulty.

We can now review the progress KPKEG has made toward its aim of establishing PSs as a viable architecture for chess programming, especially in comparison with Lisp and other conventional architectures. The standard variety of search in a tree of moves has been readily implemented, using knowledge in Ps to significantly reduce the amount of search. Modular sets of Ps cooperate smoothly to achieve an overall organization similar to a subroutine hierarchy, but with more flexibility and openness than subroutines. PSs are a concise and easily augmentable way of representing strategic knowledge in chess. PSs are also appropriate for complex selections and behavior that frequently requires complex choices. The present implementation has been useful as a pilot study of the K-P-K task, lending itself to explorations of various options and to development of control knowledge incrementally. Explorations of options take place usually by simple modifications in RHSs of Ps and by splitting an existing P into two or more finer discriminations, for action alternatives. The PS approach shows significant promise for bottom-up action, i.e., action intimately connected to the immediate problem-solving situation, which seems desirable in comparison to top-down hierarchically-controlled direction of action. There is the possibility of syntactic modifications to improve efficiency and smoothness of expression of chess patterns. Finally, approaches to more complex chess tasks are well within current PS capabilities, with natural and immediate application to several proposed mechanisms for improving the state of chess programming technology.

F.1. Acknowledgements

The author gratefully acknowledges help from Hans Berliner and Crispin Perdue in developing the concepts on which KPKEG is based, and in reading and commenting on a preliminary draft of this chapter.

G. References

Berliner, H. J., 1973. "Some necessary conditions for a master chess program", *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pp. 77-85.

Berliner, H. J., 1975a. "A representation and some mechanisms for a problem solving chess program", Pittsburgh, PA: Carnegie-Mellon University, Computer Science Department. Presented at the Second Computer Chess Conference, Oxford, England, March 1975.

Berliner, H. J., 1975b. Lectures and direct communications.

Fine, R., 1941. *Basic Chess Endings*, New York, NY: David McKay Co. Fourth Edition.

Perdue, C., 1975. Lectures and direct communications.

KPKEG APPENDICES

Appendix A. PROGRAM LISTING FOR KPKEG

```
BEGIN          % PS FOR KING & PAWN VS KING CHESS ENDGAME %

    % GROUPS OF PRODUCTIONS:
        S      STRATEGY EXECUTIVE
        Q      MAKE MOVES, UPDATING THE BOARD
        M      MEANS: GENERATE MOVES FOR STRATEGIES
        W      WHITE STRATEGIES: EVOKE MEANS
        B      BLACK STRATEGIES: EVOKE MEANS
        X      EXAMPLES FOR TESTING
    %
    % ARGUMENT TYPES:
        S      SQUARE        R      RANK
        F      FILE          P      PLAYER
        L      LEVEL         C      COLOR
        D      DEPTH         A      ACTOR = PIECE
    %
    %      MACROS:
    LEXLE(A,B) - A PREDICATE TO TEST LEXICALLY LESS THAN OR EQUAL
    LEXLT(A,B) - A PREDICATE TO TEST LEXICALLY STRICTLY LESS THAN
    PRINTBOARD(AL,H,L) - PRINT BOARD, PIECES IN ASSOC LIST AL,
                    H IS A HISTORY LIST, INDENT LEVEL L
    TRACEPRINTM(MSG,L) - PRINT MESSAGE MSG, INDENT LEVEL L
    TRACEPRINTMV(<ARGS>) - PRINT A MOVE, WITH <ARGS> = PIECE BEING
                    MOVED, SQUARES, NODE COUNT, INDENT LEVEL
    %

EXPR KPKEG(); BEGIN                          % PAGE 2 %

    DCMD(KPKC); PSMACRO(KPKEGM); REQUIRE(KPKQ,KPKM,KPKW,KPKB,KPKX);

    % STRATEGY EXECUTIVE %

S0: "INIT" : KPKINIT(X) & ISPAWN(A) & HASCOLOR(A,P)
   -> KPKHASP(P) & MAXSLEVEL('B,7) & MAXSLEVEL('W,8) & MAXDEPTH(9)
      & PLAYER('B) & PLAYER('W) & MINSLEVEL('B,1) & MINSLEVEL('W,1)
      & LASTPN('KPKW);

S1: "TOP FIND" : FINDMOVE(P) & MAXSLEVEL(P,L)
   -> PRINTBOARD('T) & CHECK:TERM(1,P) & ERSCHECK:TERM(1,P) & DEPTH(1)
      & CURLEVEL(1,L) & MOVER(P) & NEGATE(1) & MOVEHIST('(T))
      & NODECOUNT(1);

S3: "CHECK OTHERS" : CHECKOTHERSTRAT(D,P) & DEPTH(D) & NOT SELECTSTRAT(D,P)
      & NOT( EXISTS(S1,S2) & SUCCEED(D,S1,S2) )
      & NOT( EXISTS(S1,S2) & MOVECAND(D,S1,S2) )
      & NOT( EXISTS(D2) & REFUTED(D2) & SATISFIES2(D2,D,D2 EQ D - 1) )
   -> SELECTSTRAT(D,P) & CHECKOTHERSTRAT(D,P);
S4: "SELECT-" : CHECKOTHERSTRAT(D,P) & DEPTH(D)
      & NOT( NOT SELECTSTRAT(D,P) & NOT SELECTSTRAT(D,P) )
      % MAKES SELECTSTRAT LOCALLY NON-FLUENT %
   -> ERSSTRAT:TRIED(D) & CHANGELEVEL(D) & NOT SELECTSTRAT(D,P) & NEGATE(1);

S5: "DESCEND" : DESCEND(S1,S2) & DEPTH(D) & SATISFIES(D,D EQ 1) & MOVER(P1)
      & PLAYER(P2) & VNEQ(P1,P2) & MOVEHIST(X) & CURLEVEL(D,L)
   -> MAKEMOVE:T(S1,S2) & CHECK:TERM(D + 1,P2) & ERSCHECK:TERM(D + 1,P2)
      & MOVER(P2) & CURLEVEL(D + 1,L) & DEPTH(D + 1) & NEGATE(1,2,4)
      & MOVEHIST(RPLACD(X,'(S1,S2) CONS CDR X));
S6: "DESCEND2" : DESCEND(S1,S2) & DEPTH(D) & SATISFIES(D,D ?GREAT 1)
      & MOVER(P1) & PLAYER(P2) & VNEQ(P1,P2) & MOVEHIST(X) & CURLEVEL(D2,L)
      & SATISFIES2(D,D2,D2 EQ D - 1)
   -> MAKEMOVE:T(S1,S2) & CHECK:TERM(D + 1,P2) & ERSCHECK:TERM(D + 1,P2)
      & MOVER(P2) & CURLEVEL(D + 1,L) & DEPTH(D + 1) & NEGATE(1,2,8)
      & MOVEHIST(RPLACD(X,'(S1,S2) CONS CDR X));
S7: "ASCEND" : ASCEND(S1,S2) & DEPTH(D) & CURLEVEL(D,L) & MOVER(P1)
      & PLAYER(P1) & PLAYER(P2) & VNEQ(P1,P2) & MOVEHIST(X)
   -> ERSSTRAT:TRIED(D) & RETRACTMOVE(S1,S2) & RESTORE:CAP(D) & DEPTH(D - 1)
      & MOVER(P2) & NEGATE(ALL,-5,-6) & NOT SELECTSTRAT(D,P1)
      & NOT CHECKOTHERSTRAT(D,P1) & NOT CHECK:TERM(D,P1)
      & NOT ERSCHECK:TERM(D,P1) & MOVEHIST(RPLACD(X,CDDR X));
   % ASCEND AT DEPTH 1 IS IMPOSSIBLE - ASCEND ALWAYS DRIVEN BY RESULT OF MOVE %
S7E: "ERS TRIED" : ERSSTRAT:TRIED(D) & STRAT:TRIED(X,L,D) -> NEGATE(ALL);
S7F: "ERS TRIED-" : ERSSTRAT:TRIED(D)
      & NOT( EXISTS(X,L) & STRAT:TRIED(X,L,D) )
   -> NEGATE(1);
S8: "RESTORE CAP" : RESTORE:CAP(D) & CAPTURED(A,S,D)
   -> RESTORE:CON(A,D) & LOC(A,S) & NEGATE(1,2);
S8C: "RESTORE CON" : RESTORE:CON(A,D) & CONTROLLED(A,D,S2)
   -> CONTROLS(A,S2) & NEGATE(ALL);
S9: "RESTORE CAP-" : RESTORE:CAP(D) & NOT( EXISTS(S,A) & CAPTURED(A,S,D) )
   -> NEGATE(1);

S11: "SUCCESS" : SUCCEED(D,S1,S2) & SATISFIES(D,D EQ 1) & LOC(A,S1)
      & HASCOLOR(A,P)
   -> MOVING(P,A,S1,S2) & NEGATE(1);
S12: "BACK UP" : SUCCEED(D,S1,S2) & SATISFIES(D,D ?GREAT 1)
      % DEPTH IS ACTUALLY AT D + 1 HERE %
   -> ASCEND(S1,S2) & REFUTED(D - 1) & NEGATE(1);
S15: "DECR LEVEL" : CHANGELEVEL(D) & DEPTH(D)
      & SATISFIES(D,D EQ 1) & CURLEVEL(D,L) & MOVER(P)
      & MINSLEVEL(P,L2) & SATISFIES2(L,L2,L ?GREAT L2)
   -> SELECTSTRAT(D,P) & CHECKOTHERSTRAT(D,P) & CURLEVEL(D,L - 1)
      & TRACING(TRACEPRINTM('(LEVEL,?-,L - 1,P),D)) & NEGATE(1,4);
S16: "DECR LEVEL-" : CHANGELEVEL(D) & DEPTH(D)
      & SATISFIES(D,D EQ 1) & CURLEVEL(D,L) & MOVER(P) & MINSLEVEL(P,L)
   -> DEPTH(D) & TRACING(TRACEPRINTM('(LEVEL,?-,FAIL,DEPTH,D,P),D))
      & NEGATE(1);
S17: "INCR LEVEL" : CHANGELEVEL(D) & DEPTH(D)
      & SATISFIES(D,D ?GREAT 1) & CURLEVEL(D,L) & MOVER(P)
      & MAXSLEVEL(P,L2) & SATISFIES2(L,L2,L ?LESS L2)
   -> SELECTSTRAT(D,P) & CHECKOTHERSTRAT(D,P) & CURLEVEL(D,L + 1)
      & TRACING(TRACEPRINTM('(LEVEL,?-,L + 1,P),D)) & NEGATE(1,4);
S18: "INCR LEVEL-" : CHANGELEVEL(D) & DEPTH(D)
      & SATISFIES(D,D ?GREAT 1) & CURLEVEL(D,L) & MOVER(P) & MAXSLEVEL(P,L)
   -> DEPTH(D) & TRACING(TRACEPRINTM('(LEVEL,?-,FAIL,DEPTH,D,P),D))
      & NEGATE(1);

S21: "SELECT MOVE" : MOVECAND(D,S1,S2) & DEPTH(D)
      & NOT( EXISTS(S3,S4) & CHECKMOVERESULT(D,S3,S4) )
      & NOT( EXISTS(S3,S4) & MOVECAND(D,S3,S4) & WINCAND(D,S3,S4) )
      % ASSUMING MUST WAIT TILL MOVECAND GEN'D BEFORE WINCAND TAKES %
      & ISPAWN(A) & LOC(A,S) & RF(S,R,F) & RF(S2,R2,F2)
      % THAT RF EXCLUDES OFFBOARD S2'S %
      & NOT( EXISTS(C,A1,A2) & LOC(A1,S1) & LOC(A2,S2)
            & HASCOLOR(A1,C) & HASCOLOR(A2,C) )
      & NOT( EXISTS(S3,S4,R4,F4) & MOVECAND(D,S3,S4) & RF(S4,R4,F4)
            & SATISFIES2(R4,F4,MAX(ABS(R-R4),ABS(F-F4))
                      ?LESS MAX(ABS(R-R2),ABS(F-F2))) )
            % DISTANCE TO PAWN'S Q SQUARE IS LESS %
      & NOT( EXISTS(S3,S4,R4,F4) & MOVECAND(D,S3,S4) & RF(S4,R4,F4)
            & SATISFIES2(R4,F4,MAX(ABS(R-R4),ABS(F-F4))
                      EQ MAX(ABS(R-R2),ABS(F-F2)))
            & SATISFIES2(R4,F4,MIN(ABS(R-R4),ABS(F-F4))
                      ?LESS MIN(ABS(R-R2),ABS(F-F2))) )
      & NOT( EXISTS(S3,S4,R4,F4) & MOVECAND(D,S3,S4) & RF(S4,R4,F4)
            & SATISFIES2(R4,F4,MAX(ABS(R-R4),ABS(F-F4))
                      EQ MAX(ABS(R-R2),ABS(F-F2)))
            & SATISFIES2(R4,F4,MIN(ABS(R-R4),ABS(F-F4))
                      EQ MIN(ABS(R-R2),ABS(F-F2)))
            & SATISFIES2(S4,S2,S4 LEXLT S2) )
      & NOT( EXISTS(S3) & MOVECAND(D,S3,S2) & SATISFIES2(S1,S3,S3 LEXLT S1) )
      % PICKS DESTINATION SQUARE CLOSEST TO PAWN'S QUEENING SQUARE %
      % AMONG EQUALS BY THAT, CLOSEST BY MIN ALSO, THEN LEXLT'ST DEST,
            THEN LEXLT'ST SOURCE WITH THAT UNIQUE DEST %
   -> DESCEND(S1,S2) & CHECKMOVERESULT(D,S1,S2) & NEGATE(1);
S21A: "SELECT WIN" : MOVECAND(D,S1,S2) & WINCAND(D,S1,S2) & DEPTH(D)
      % CAN ONLY BE ONE SUCH WINCAND %
   -> DESCEND(S1,S2) & CHECKMOVERESULT(D,S1,S2) & NEGATE(1,2)
      & TRACING(TRACEPRINTM('(USING,WINCAND,S2,1,A),D));
S21D: "SELECT OWN" : MOVECAND(D,S1,S2) & DEPTH(D) & LOC(A1,S1) & LOC(A2,S2)
      & HASCOLOR(A1,C) & HASCOLOR(A2,C)
   -> DEPTH(D) & NEGATE(1);
S21O: "SELECT OFF" : MOVECAND(D,S1,S2) & DEPTH(D) & OFFBOARD(S2)
   -> DEPTH(D) & NEGATE(1);
S23: "NOT REFUTED" : CHECKMOVERESULT(D,S1,S2) & NOT REFUTED(D) & DEPTH(D2)
      & SATISFIES2(D,D2,D2 EQ D + 1)
      & NOT( EXISTS(S3,S4) & MOVECAND(D2,S3,S4) )
      & NOT( EXISTS(P) & CHECKOTHERSTRAT(D2,P) )
   -> PRINTBOARD('T) & ERSMOVES(D) & RECORDWIN(D,S1,S2) & SUCCEED(D,S1,S2)
      & TRACING(TRACEPRINTM('(SUCCEED,S1,S2,?-,S2,3),D2)) & NEGATE(1);
S24: "SUCC STRAT" : SUCCSTRAT(D,P,L,X)
   -> PRINTBOARD('T) & REFUTED(D - 1) & NEGATE(1)
      & TRACING(TRACEPRINTM('(SUCCEED,STRAT,LEVEL,L,?-,X),D));
S25: "REFUTED" : CHECKMOVERESULT(D,S1,S2) & REFUTED(D) & DEPTH(D2)
      & SATISFIES2(D,D2,D2 EQ D + 1)
   -> ASCEND(S1,S2) & NEGATE(1,2);     % FALL BACK %
S26: "ERS MOVES" : ERSMOVES(D) & MOVECAND(D,S1,S2)
   -> NEGATE(ALL);
S26N: "ERS MOVES-" : ERSMOVES(D) & NOT( EXISTS(S1,S2) & MOVECAND(D,S1,S2) )
   -> NEGATE(1);

S30: "MAX DEPTH" : CHECK:TERM(D,P) & MAXDEPTH(D) & NOT STATICEVAL(D,P)
   -> CHECK:TERM(D,P) & STATICEVAL(D,P);
S31: "WIN W" : CHECK:TERM(D,P) & ISPAWN(A) & LOC(A,S) & KPKHASP(P)
      & RF(S,R,F) & SATISFIES(R,R EQ 8) & ISKING(A2) & HASCOLOR(A2,C)
      & SATISFIES(C,C EQ 'B) & ISKING(A3) & VNEQ(A2,A3)
      & NOT( CONTROLS(A2,S) & NOT CONTROLS(A3,S) )
      & LOC(A2,S2) & NOT( SATISFIES(S,S EQ 'C8) & SATISFIES(S2,S2 EQ 'A7) )
      & NOT( SATISFIES(S,S EQ 'F8) & SATISFIES(S2,S2 EQ 'H7) )
   -> TERMWIN(P,'S31) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);
S32: "LOSE W" = CHECK:TERM(D,P) & NOT( EXISTS(A,S) & ISPAWN(A) & LOC(A,S) )
      & PLAYER(P2) & NOT KPKHASP(P2) & KPKHASP(P3)
   -> TERMWIN(P2,'S32) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);
S33: "STALE" = CHECK:TERM(D,C) & NOT KPKHASP(C) & ISKING(A) & HASCOLOR(A,C)
      & LOC(A,S) & KPKHASP(P)
      & NOT( EXISTS(A2,C7) & CONTROLS(A2,S) & HASCOLOR(A2,C7) & VNEQ(C,C7) )
      & NOT( EXISTS(S2) & CONTROLS(A,S2) & NOT( EXISTS(A7) & LOC(A7,S2) )
            & NOT OFFBOARD(S2)
            & NOT( EXISTS(A2,C7) & CONTROLS(A2,S2) & HASCOLOR(A2,C7)
                  & VNEQ(C,C7) ) )
   -> TERMWIN(C,'S33) & NEGATE(1) & NOT ERSCHECK:TERM(D,C);
S34: "MATE" = CHECK:TERM(D,C) & ISKING(A) & LOC(A,S) & CONTROLS(A2,S)
      & HASCOLOR(A,C) & HASCOLOR(A2,C2) & VNEQ(C,C7)
      & NOT( EXISTS(S2) & CONTROLS(A,S2) & NOT( EXISTS(A3) & LOC(A3,S2) )
            & NOT OFFBOARD(S2)
            & NOT( EXISTS(A3) & CONTROLS(A3,S2) & HASCOLOR(A3,C7) ) )
   -> TERMWIN(C7,'S34) & NEGATE(1) & NOT ERSCHECK:TERM(D,C);
S35: "BK OPPOS" = CHECK:TERM(D,P) & NOT SATISFIES(D,D EQ 1) & KPKHASP(P)
      & ISKING(A1) & HASCOLOR(A1,P7) & VNEQ(P,P2) & ISKING(A2)
      & HASCOLOR(A2,P) & LOC(A1,S1) & FILEB(S1,S7) & FILEB(S7,S3) & LOC(A2,S3)
      & ISPAWN(A3) & RF(S3,R1,F1) & LOC(A3,S4) & RF(S4,R2,F2)
      & VNEQ(F1,F2)   % BK HAS OPPOS AND WK NOT IN FRONT OF WP DIRECTLY %
      & NOT( SATISFIES(R1,R1 EQ 6) & SATISFIES2(R1,R2,R1 ?>GREAT R2) )
      % WK NOT( ON SIXTH RANK & IN FRONT OF WP ) %
   -> TERMWIN(P2,'S35) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);
S36: "WK FRONT2+" = CHECK:TERM(D,P) & NOT SATISFIES(D,D EQ 1)
      & KPKHASP(C1) & ISKING(A1) & HASCOLOR(A1,C1) & ISKING(A2)
      & VNEQ(A2,A1) & ISPAWN(A3)
      & LOC(A3,S1) & RF(S1,R1,F1) & LOC(A1,S7) & RF(S2,R2,F1)
      & SATISFIES2(R1,R2,R2 ?>GREAT R1+1)
      & LOC(A2,S3) & RF(S3,R3,F3)
      & NOT SATISFIES3(R2,R3,F3,MAX(ABS(R3-R1),ABS(F3-F1)) ?<LESS R2-R1)
      % WK DIR IN FRONT OF P, BK NOT CLOSER TO P %
   -> TERMWIN(C1,'S36) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);
S360: "WK FRONT1" = CHECK:TERM(D,P) & NOT SATISFIES(D,D EQ 1) & NOT KPKHASP(P)
      & ISKING(A1) & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A1,A2) & KPKHASP(C)
      & HASCOLOR(A2,P2) & ISPAWN(A3) & LOC(A3,S1)
      & LOC(A2,S2) & FILEF(S1,S7) & FILEF(S7,S3) & FILEF(S3,S4) & LOC(A1,S4)
      % WK DIR IN FRONT OF WP AND HAS OPPOS %
   -> TERMWIN(P2,'S360) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);
S36R: "WK FRONT 6" = CHECK:TERM(D,P) & NOT SATISFIES(D,D EQ 1) & ISKING(A1)
      & HASCOLOR(A1,C1) & KPKHASP(C1) & ISPAWN(A2)
      & LOC(A1,S1) & LOC(A2,S2) & RF(S1,R1,F1) & SATISFIES(R1,R1 EQ 6)
      & RF(S2,R2,F2) & SATISFIES2(R1,R2,R1 ?>GREAT R2 & R1 ?<LESS R2+3)
      & SATISFIES2(F1,F2,ABS(F1-F2) ?<LESS 2)
      & ISKING(A3) & VNEQ(A1,A3) & LOC(A3,S3) & RF(S3,R3,F3)
      & NOT SATISFIES3(R1,R2,R3,MAX(ABS(R3-R2),ABS(F3-F2))
            ?<LESS MAX(ABS(R1-R2),ABS(F1-F2)))
      % WK ON SIXTH RANK IN FRONT OF P, BK NOT CLOSER TO P %
   -> TERMWIN(C1,'S36R) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);
S37L: "STALE Q L" = CHECK:TERM(D,P) & NOT KPKHASP(P) & ISPAWN(A1)
      & KPKHASP(C)
      & HASCOLOR(A1,P2) & VNEQ(P2,P) & LOC(A1,S1) & SATISFIES(S1,S1 EQ 'C8)
      & ISKING(A7) & HASCOLOR(A2,P) & LOC(A7,S7) & SATISFIES(S2,S2 EQ 'A7)
      & ISKING(A3) & VNEQ(A3,A7) & CONTROLS(A3,S3) & SATISFIES(S3,S3 EQ 'B6)
   -> TERMWIN(P,'S37L) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);
S37R: "STALE Q R" = CHECK:TERM(D,P) & NOT KPKHASP(P) & ISPAWN(A1)
      & KPKHASP(C)
      & HASCOLOR(A1,P2) & VNEQ(P2,P) & LOC(A1,S1) & SATISFIES(S1,S1 EQ 'F8)
      & ISKING(A7) & HASCOLOR(A7,P) & LOC(A7,S7) & SATISFIES(S2,S2 EQ 'H7)
      & ISKING(A3) & VNEQ(A3,A7) & CONTROLS(A3,S3) & SATISFIES(S3,S3 EQ 'G6)
   -> TERMWIN(P,'S37R) & NEGATE(1) & NOT ERSCHECK:TERM(D,P);

S38: "ERS CHK" = ERSCHECK:TERM(D,P)
   -> SELECTSTRAT(D,P) & CHECKOTHERSTRAT(D,P) & NEGATE(1)
      & NOT CHECK:TERM(D,P);
S39: "STATIC EVAL" = ERSCHECK:TERM(D,P) & STATICEVAL(D,P)
   -> SELECTSTATIC(D,P) & NEGATE(1,2) & NOT CHECK:TERM(D,P);
   % IF NO POSITIVE RESPONSE TO SELECTSTATIC, PREVIOUS DEPTH IS NOT
      REFUTED %

S41: "TERM RES." = TERMWIN(P,X) & MOVER(P) & DEPTH(D)
   -> PRINTBOARD('T) & REFUTED(D - 1) & NEGATE(1) & NOT STATICEVAL(D,P)
      & TRACING(TRACEPRINTM(<TERMINAL WIN FOR P ?,X>D));
S42: "TERM RES." = TERMWIN(P,X) & MOVER(P2) & VNEQ(P,P2) & DEPTH(D)
      & SATISFIES(D,D ?>GREAT 1) & MAXSLEVEL(P2,L) & CURLEVEL(D,L,2)
   -> PRINTBOARD('T) & CHECKOTHERSTRAT(D,P2) & NEGATE(1,7) & CURLEVEL(D,L)
      & NOT STATICEVAL(D,P2)
      & TRACING(TRACEPRINTM(<TERMINAL WIN FOR P ?,X>D));
S43: "TERM RES-1" = TERMWIN(P,X) & MOVER(P2) & VNEQ(P,P2) & DEPTH(D)
      & SATISFIES(D,D EQ 1) & MINSLEVEL(P2,L) & CURLEVEL(D,L,2)
   -> CHECKOTHERSTRAT(D,P2) & NEGATE(1,7) & CURLEVEL(D,L)
      & NOT STATICEVAL(D,P2)
      & TRACING(TRACEPRINTM(<TERMINAL WIN FOR P ?,X>D));

S50: "P BOARD" = PRINTBOARD(X) & NOT PRINTEDBOARD(X) & KPKHASP(P) & DEPTH(D)
      & ISPAWN(A1) & LOC(A1,S1) & ISKING(A2) & LOC(A2,S2) & ISKING(A3)
      & SATISFIES2(A3,A2,A2 LEXLT A3) & LOC(A3,S3) & RF(S1,R1,F1)
      & RF(S2,R2,F2) & RF(S3,R3,F3) & MOVEHIST(Z)
   -> TRACING(PRINTBOARD(<R1,F1,A1>,<R2,F2,A2>,<R3,F3,A3>),Z,D))
      & PRINTEDBOARD('T) & NEGATE(1));
S51: "P BOARD" = PRINTBOARD(X) & NOT PRINTEDBOARD(X) & KPKHASP(P) & DEPTH(D)
      & NOT( EXISTS(A1,S1) & ISPAWN(A1) & LOC(A1,S1) )
      & ISKING(A2) & LOC(A2,S2) & ISKING(A3) & SATISFIES2(A3,A2,A2 LEXLT A3)
      & LOC(A3,S3)  & RF(S2,R2,F2) & RF(S3,R3,F3) & MOVEHIST(Z)
   -> TRACING(PRINTBOARD(<R2,F2,A2>,<R3,F3,A3>),Z,D))
      & PRINTEDBOARD('T) & NEGATE(1));
S52: "P BOARD-" = PRINTBOARD(X) & PRINTEDBOARD(X) -> NEGATE(1);

S80: "REC P P" = RECORDWIN(D,S4,S1) & CURLEVEL(D,L) & KPKHASP(P1)
      & ISPAWN(A1) & LOC(A1,S1) & ISKING(A2) & HASCOLOR(A2,P1) & ISKING(A3)
      & VNEQ(A3,A2) & LOC(A2,S7) & LOC(A3,S3)
   -> RECORDPRE(D,L,S4,S1,S4,S2,S3,P1) & NEGATE(1);
S81: "REC P WK" = RECORDWIN(D,S4,S1) & CURLEVEL(D,L) & KPKHASP(P1)
      & ISKING(A1) & LOC(A1,S1) & HASCOLOR(A1,P1) & ISPAWN(A2) & LOC(A2,S2)
      & ISKING(A3) & VNEQ(A3,A1) & LOC(A3,S3)
   -> RECORDPRE(D,L,S4,S1,S7,S4,S3,P1) & NEGATE(1);
S82: "REC P BK" = RECORDWIN(D,S4,S1) & CURLEVEL(D,L) & KPKHASP(P1)
      & ISKING(A1) & LOC(A1,S1) & HASCOLOR(A1,P2) & VNEQ(P2,P1)
      & ISPAWN(A2) & LOC(A2,S2) & ISKING(A3) & VNEQ(A3,A1) & LOC(A3,S3)
   -> RECORDPRE(D,L,S4,S1,S7,S3,S4,P2) & NEGATE(1);
S83: "REC PRE" = RECORDPRE(D,L,S4,S5,S1,S2,S3,P)
   -> RECORDOLD(D,L,S4,S5,<<CHECK:TERM,'D,'P>,
            <SATISFIES,'P,<EQ,'P,<QUOTE,P>>>,
            <ISPAWN,'A1>,<LOC,'A1,'S1>,<SATISFIES,'S1,
                  <EQ,'S1,<QUOTE,S1>>>,
            <ISKING,'A2>,<LOC,'A2,'S2>,<SATISFIES,'S2,
                  <EQ,'S2,<QUOTE,S2>>>,
            <HASCOLOR,'A2,'P2>,<KPKHASP,'P2>,<ISKING,'A3>,
            <LOC,'A3,'S3>,<SATISFIES,'S3,<EQ,'S3,<QUOTE,S3>>> >)
      & RECORDFIN(D) & NEGATE(1);
S84: "REC SUC" = RECORDOLD(D1,L1,S4,S5,X) & LASTPN(Y)
      & NOT( EXISTS(Z) & RECORDDONE(D1,Z) & SATISFIES(Z,Z EQ 'PN) )
   -> EXISTS(PN) & ADDPROD(PN,Y,NIL,X,@
      % TAIL OF LHS: % <<SATISFIES,'D,<AND,<NEQ,'D,1>,
                  <NOT,<?<LESS,'D,D1>>>>,
            <CURLEVEL,'D,'L>,<SATISFIES,'L,<NOT,<?>GREAT,'L,L1>>>>,
      % RHS: % <<TERMWIN,'P,<QUOTE,PN>>,<NEGATE,1>,
                  <NOT,<ERSCHECK:TERM,'D,'P>>>)
      & LASTPN(PN) & RECORDDONE(D1,'PN) & NEGATE(2) & NOT ADDPROD(PN)
      & TRACING(TRACEPRINTM(<ADDPROD,PN,'DEPTH,D1,'LEVEL,L1,S4,S5>,D1+1));
S85: "REC WIN C 1" = RECORDOLD(D1,L1,S4,S5,X) & LASTPN(Y)
      & NOT( EXISTS(Z) & RECORDDONE(D1,Z) & SATISFIES(Z,Z EQ 'PV) )
   -> EXISTS(PV) & ADDPROD(PV,Y,NIL,X,@
      % TAIL OF LHS: % <<SATISFIES,'D,<EQ,'D,1>>,
            <CURLEVEL,'D,'L>,<SATISFIES,'L,<EQ,'L,L1>>>,
      % RHS: % <<WINCAND,'D,<QUOTE,S4>,<QUOTE,S5>>>)
      & LASTPN(PV) & RECORDDONE(D1,'PV) & NEGATE(2) & NOT ADDPROD(PV);
      % NOT POSITIVELY A WINNER, BUT RECOMMEND IT FOR SEARCH %
S86: "REC WIN C L" = RECORDOLD(D1,L1,S4,S5,X) & LASTPN(Y)
      & SATISFIES(D1,D1 ?>GREAT 2)
      & NOT( EXISTS(Z) & RECORDDONE(D1,Z) & SATISFIES(Z,Z EQ 'PW) )
   -> EXISTS(PW) & ADDPROD(PW,Y,NIL,X,@
      % TAIL OF LHS: % << SATISFIES,'D,<?<LESS,'D,D1>>
            <CURLEVEL,'D,'L>,<SATISFIES,'L,<EQ,'L,L1>>>,
      % RHS: % <<WINCAND,'D,<QUOTE,S4>,<QUOTE,S5>>>)
      & LASTPN(PW) & RECORDDONE(D1,'PW) & NEGATE(2) & NOT ADDPROD(PW);
      % NOT POSITIVELY A WINNER, BUT RECOMMEND IT FOR SEARCH %
S87: "REC FIN" = RECORDFIN(D) -> RECORDFIN2(D) & NEGATE(1);
S88: "REC FIN2" = RECORDFIN2(D) & RECORDOLD(D,L,S1,S2,Z) & RECORDDONE(D,X)
   -> NEGATE(ALL);

   END;
```

....

```
EXPR KPKQ(): BEGIN            % MAKE MOVES, UPDATING BOARD % % PAGE 3 %

S0: "TRACE" = MAKEMOVE:T(S1,S2) & LOC(A,S1) & NOT( EXISTS(A7) & LOC(A2,S2) )
```

```
      & DEPTH(D) & CURLEVEL(D,L) & NODECOUNT(X)
   -> MAKEMOVE(S1,S2) & TRACING(TRACEPRINTMV(X,A,S1,S2,NIL,L,D-1))
      & NEGATE(1,6) & NOT PRINTEDBOARD(T) & NODECOUNT(X+1);
Q0C: "TRACE CAP" = MAKEMOVE:T(S1,S2) & LOC(A1,S1) & LOC(A2,S2)
      & DEPTH(D) & CURLEVEL(D,L) & NODECOUNT(X)
   -> MAKEMOVE(S1,S2) & TRACING(TRACEPRINTMV(X,A1,S1,S2,A2,L,D-1))
      & NEGATE(1,6) & NOT PRINTEDBOARD(T) & NODECOUNT(X+1);

Q1: "RETRACT" = RETRACTMOVE(S1,S2) & NOT RETRACTHOLD(S1,S2) & DEPTH(D)
   -> MAKEMOVE(S2,S1) & TRACING(TRACEPRINTM(<RETRACTING,S1,S2>,D))
      & RETRACTING(S2,S1) & NEGATE(1);
Q2: "RETRACT-" = RETRACTMOVE(S1,S2) & RETRACTHOLD(S1,S2) & DEPTH(D)
   -> TRACING(TRACEPRINTM(<"CAN'T MOVE",S1,S2>,D)) & NEGATE(1,2);

Q3: "P FORWARD" = MAKEMOVE(S1,S2) & LOC(A,S1) & ISPAWN(A) & NOT OFFBOARD(S2)
      & NOT( EXISTS(A2,C) & LOC(A2,S2) & HASCOLOR(A,C) & HASCOLOR(A2,C) )
      & FILEF(S1,S2) & CONTROLS(A,S3) & CONTROLS(A,S4)
      & SATISFIES2(S3,S4,S3 LEXLT S4) & FILEF(S3,S5) & FILEF(S4,S6)
   -> LOC(A,S2) & CONTROLS(A,S5) & CONTROLS(A,S6) & NEGATE(1,2,7,8);
   % FORWARD IS AWAY FROM W'S HOME ROW %
Q4: "P BACK" = MAKEMOVE(S1,S2) & LOC(A,S1) & ISPAWN(A) & FILEB(S1,S2)
      & CONTROLS(A,S3) & CONTROLS(A,S4) & SATISFIES2(S3,S4,S3 LEXLT S4)
      & FILEB(S3,S5) & FILEB(S4,S6)
   -> LOC(A,S2) & CONTROLS(A,S5) & CONTROLS(A,S6) & NEGATE(1,2,5,6)
      & NOT RETRACTING(S1,S2);
   % NO CAPTURES BY P CONSIDERED HERE %

Q7: "KING CONTR" = MAKEMOVE(S1,S2) & LOC(A1,S1) & ISKING(A1) & HASCOLOR(A1,C1)
      & CONTROLS(A2,S2) & HASCOLOR(A2,C2) & VNEQ(C1,C2) & DEPTH(D)
      & NOT RETRACTING(S1,S2)
   -> REFUTED(D - 1) & RETRACTHOLD(S1,S2) & NEGATE(1);

Q8: "CHECK CAP" = CHECKCAP(A,S) & LOC(A2,S) & VNEQ(A,A2) & DEPTH(D)
   -> SAVECON(A2,D) & CAPTURED(A2,S,D) & NEGATE(1,2);
Q8C: "SAVE CON" = SAVECON(A,D) & CONTROLS(A,S)
   -> CONTROLLED(A,D,S) & NEGATE(ALL);
Q9: "CHECK CAP-" = CHECKCAP(A,S) & NOT( EXISTS(A2) & LOC(A2,S) & VNEQ(A,A2) )
   -> NEGATE(1);

Q11: "K COMMON" = MAKEMOVE(S1,S2) & LOC(A1,S1) & ISKING(A1) & NOT OFFBOARD(S2)
      & NOT( EXISTS(C1,C2,A2) & CONTROLS(A2,S2) & HASCOLOR(A1,C1)
            & HASCOLOR(A2,C2) & VNEQ(C1,C2) & NOT RETRACTING(S1,S2) )
      & NOT( EXISTS(A2,C) & LOC(A2,S2) & HASCOLOR(A1,C) & HASCOLOR(A2,C) )
      & CONTROLS(A1,S2)
   -> CHECKCAP(A1,S2) & MAKEMOVEK(A1,S1,S2) & LOC(A1,S2) & CONTROLS(A1,S1)
      & NEGATE(1,2,7) & NOT RETRACTING(S1,S2);
   % ONE CONTROLLED SQUARE ALWAYS CHANGED %
Q12: "K FORWARD" = MAKEMOVEK(A1,S1,S2) & FILEF(S1,S2)
      & FILEB(S1,S3) & CONTROLS(A1,S3)
      & DIAGLB(S1,S4) & CONTROLS(A1,S4) & DIAGRB(S1,S5) & CONTROLS(A1,S5)
      & DIAGLF(S2,S6) & FILEF(S2,S7) & DIAGRF(S2,S8)
   -> CONTROLS(A1,S6) & CONTROLS(A1,S7) & CONTROLS(A1,S8) & NEGATE(1,4,6,8);
   % 4 CONTROLLED SQUARES STAY CONTROLLED, 3 CHANGE %
Q13: "K BACK" = MAKEMOVEK(A1,S1,S2) & FILEB(S1,S2)
      & FILEF(S1,S3) & CONTROLS(A1,S3)
      & DIAGLF(S1,S4) & CONTROLS(A1,S4) & DIAGRF(S1,S5) & CONTROLS(A1,S5)
      & DIAGLB(S2,S6) & FILEB(S2,S7) & DIAGRB(S2,S8)
   -> CONTROLS(A1,S6) & CONTROLS(A1,S7) & CONTROLS(A1,S8) & NEGATE(1,4,6,8);
   % 4 CONTROLLED SQUARES STAY CONTROLLED, 3 CHANGE %
Q14: "K LEFT" = MAKEMOVEK(A1,S1,S2) & RANKL(S1,S2)
      & RANKR(S1,S3) & CONTROLS(A1,S3)
      & DIAGRF(S1,S4) & CONTROLS(A1,S4) & DIAGRB(S1,S5) & CONTROLS(A1,S5)
      & DIAGLF(S2,S6) & RANKL(S2,S7) & DIAGLB(S2,S8)
   -> CONTROLS(A1,S6) & CONTROLS(A1,S7) & CONTROLS(A1,S8) & NEGATE(1,4,6,8);
   % 4 CONTROLLED SQUARES STAY CONTROLLED, 3 CHANGE %
Q15: "K RIGHT" = MAKEMOVEK(A1,S1,S2) & RANKR(S1,S2)
      & RANKL(S1,S3) & CONTROLS(A1,S3)
      & DIAGLF(S1,S4) & CONTROLS(A1,S4) & DIAGLB(S1,S5) & CONTROLS(A1,S5)
      & DIAGRF(S2,S6) & RANKR(S2,S7) & DIAGRB(S2,S8)
   -> CONTROLS(A1,S6) & CONTROLS(A1,S7) & CONTROLS(A1,S8) & NEGATE(1,4,6,8);
   % 4 CONTROLLED SQUARES STAY CONTROLLED, 3 CHANGE %
Q16: "K DIAGLF" = MAKEMOVEK(A1,S1,S2) & DIAGLF(S1,S2)
      & DIAGRF(S1,S3) & CONTROLS(A1,S3) & DIAGRB(S1,S4)
      & CONTROLS(A1,S4) & RANKR(S1,S5) & CONTROLS(A1,S5) & FILEB(S1,S6)
      & CONTROLS(A1,S6) & DIAGLB(S1,S7) & CONTROLS(A1,S7)
      & DIAGRF(S2,S8) & DIAGLF(S2,S9) & DIAGLB(S2,S10) & FILEF(S2,S11)
      & RANKL(S2,S12)
   -> CONTROLS(A1,S8) & CONTROLS(A1,S9) & CONTROLS(A1,S10) & CONTROLS(A1,S11)
      & CONTROLS(A1,S12) & NEGATE(1,4,6,8,10,12);
   % 5 CONTROLS CHANGED, 2 THE SAME %
Q17: "K DIAGRF" = MAKEMOVEK(A1,S1,S2) & DIAGRF(S1,S2)
      & DIAGLF(S1,S3) & CONTROLS(A1,S3) & DIAGLB(S1,S4)
      & CONTROLS(A1,S4) & RANKL(S1,S5) & CONTROLS(A1,S5) & FILEB(S1,S6)
      & CONTROLS(A1,S6) & DIAGRB(S1,S7) & CONTROLS(A1,S7)
      & DIAGLF(S2,S8) & DIAGRF(S2,S9) & DIAGRB(S2,S10) & FILEF(S2,S11)
      & RANKR(S2,S12)
   -> CONTROLS(A1,S8) & CONTROLS(A1,S9) & CONTROLS(A1,S10) & CONTROLS(A1,S11)
      & CONTROLS(A1,S12) & NEGATE(1,4,6,8,10,12);
   % 5 CONTROLS CHANGED, 2 THE SAME %
Q18: "K DIAGLB" = MAKEMOVEK(A1,S1,S2) & DIAGLB(S1,S2)
      & DIAGRB(S1,S3) & CONTROLS(A1,S3) & DIAGRF(S1,S4)
      & CONTROLS(A1,S4) & RANKR(S1,S5) & CONTROLS(A1,S5) & FILEF(S1,S6)
      & CONTROLS(A1,S6) & DIAGLF(S1,S7) & CONTROLS(A1,S7)
      & DIAGRB(S2,S8) & DIAGLB(S2,S9) & DIAGLF(S2,S10) & FILEB(S2,S11)
      & RANKL(S2,S12)
   -> CONTROLS(A1,S8) & CONTROLS(A1,S9) & CONTROLS(A1,S10) & CONTROLS(A1,S11)
      & CONTROLS(A1,S12) & NEGATE(1,4,6,8,10,12);
   % 5 CONTROLS CHANGED, 2 THE SAME %
Q19: "K DIAGRB" = MAKEMOVEK(A1,S1,S2) & DIAGRB(S1,S2)
      & DIAGLB(S1,S3) & CONTROLS(A1,S3) & DIAGLF(S1,S4)
      & CONTROLS(A1,S4) & RANKL(S1,S5) & CONTROLS(A1,S5) & FILEF(S1,S6)
      & CONTROLS(A1,S6) & DIAGRF(S1,S7) & CONTROLS(A1,S7)
      & DIAGLB(S2,S8) & DIAGRB(S2,S9) & DIAGRF(S2,S10) & FILEB(S2,S11)
      & RANKR(S2,S12)
   -> CONTROLS(A1,S8) & CONTROLS(A1,S9) & CONTROLS(A1,S10) & CONTROLS(A1,S11)
      & CONTROLS(A1,S12) & NEGATE(1,4,6,8,10,12);
   % 5 CONTROLS CHANGED, 2 THE SAME %

   END;

                        ......................


EXPR KPMOVE): BEGIN     % MEANS TO STRATEGIES - MOVE GEN'S %    % PAGE 6 %

M1: "MOVE TW DRB" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R2,F2) & SATISFIES2(F1,F2,F1 %LESS F2)
      & SATISFIES2(R1,R2,R1 %GREAT R2)
      & RANKR(S1,S3) & FILEB(S1,S4) & DIAGRB(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);
M2: "MOVE TW B" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R2,F1) & SATISFIES2(R1,R2,R1 %GREAT R2)
      & DIAGLB(S1,S3) & FILEB(S1,S4) & DIAGRB(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);
M3: "MOVE TW DLB" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R2,F2) & SATISFIES2(F1,F2,F1 %GREAT F2)
      & SATISFIES2(R1,R2,R1 %GREAT R2)
      & RANKL(S1,S3) & FILEB(S1,S4) & DIAGLB(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);
M4: "MOVE TW R" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R1,F2) & SATISFIES2(F1,F2,F1 %LESS F2)
      & RANKR(S1,S3) & DIAGRF(S1,S4) & DIAGRB(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);
M5: "MOVE TW L" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R1,F2) & SATISFIES2(F1,F2,F1 %GREAT F2)
      & RANKL(S1,S3) & DIAGLF(S1,S4) & DIAGLB(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);
M6: "MOVE TW DRF" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R2,F2) & SATISFIES2(F1,F2,F1 %LESS F2)
      & SATISFIES2(R1,R2,R1 %LESS R2)
      & RANKR(S1,S3) & FILEF(S1,S4) & DIAGRF(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);
M7: "MOVE TW F" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R2,F1) & SATISFIES2(R1,R2,R1 %LESS R2)
      & DIAGLF(S1,S3) & FILEF(S1,S4) & DIAGRF(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);
M8: "MOVE TW DLF" = MOVE:TOWARD(D,A,S2) & NOT CONTROLS(A,S2) & LOC(A,S1)
      & RF(S1,R1,F1) & RF(S2,R2,F2) & SATISFIES2(F1,F2,F1 %GREAT F2)
      & SATISFIES2(R1,R2,R1 %LESS R2)
      & RANKL(S1,S3) & FILEF(S1,S4) & DIAGLF(S1,S5)
   -> MOVEHOLD(D,S1,S3) & MOVEHOLD(D,S1,S4) & MOVEHOLD(D,S1,S5) & NEGATE(1);

M9: "MOVE TW TO" = MOVE:TOWARD(D,A,S1) & CONTROLS(A,S1) & LOC(A,S2)
      & NOT OFFBOARD(S1)
   -> MOVEHOLD(D,S2,S1) & NEGATE(1);
M9F: "MOVE TW-" = MOVE:TOWARD(D,A,S1) & OFFBOARD(S1) -> NEGATE(1);
M9N: "MOVE TW ON" = MOVE:TOWARD(D,A,S1) & LOC(A,S1) -> NEGATE(1);

M11: "HOLD-" = MOVEHOLD(D,S1,S2) & NOT MEANSHOLD(D)
   -> MOVECAND(D,S1,S2) & NEGATE(1);
M12: "HOLD+" = MEANSHOLD(D) -> MEANSRELS(D) & NEGATE(1);
M13: "RELS" = MEANSRELS(D) & MOVEHOLD(D,S1,S2) & NOT MEANSEXAM(D)
   -> MOVECAND(D,S1,S2) & NEGATE(1,2);
M13X: "RELS X" = MEANSRELS(D) & MEANSEXAM(D) & MOVEHOLD(D,S1,S2)
   -> MOVEEXAM(D,S1,S2) & NEGATE(ALL);
```

```
M14: "RELS-" = MEANSRELS(D) & NOT( EXISTS(S1,S2) & MOVEHOLD(D,S1,S2) )
   -> NEGATE(1);

M16: "MOVE TO-" = MOVE:TO(D,A,S1) & CONTROLS(A,S1) & LOC(A,S2)
   -> MOVEHOLD(D,S2,S1) & NEGATE(1);
M17: "MOVE TO-" = MOVE:TO(D,A,S) & NOT CONTROLS(A,S) -> NEGATE(1);

   END;


                     - - - - - - - - - - - - - - - - - - - -


EXPR KPKW(): BEGIN     % WHITE STRATEGIES %          % PAGE 8 %

% STRATEGIES FOR:      W                   B
    LEVELS
      7      MATE                  CAPTURE P
      6      QUEEN P                STALEMATE
      5      ADVANCE P                INTERCEPT P, STAY IN SQUARE
      4      CONTROL PATH OF P      OCCUPY PATH OF P
      3      DEFEND P               ATTACK UNPROTECTED P
      2      RESTRICT K MOVES        SAME
      1      SOME MOVE UNLIKE ABOVE  SAME
             OPP. OF PRECEDING

 %


      % IN KPK, MATE IMPOSSIBLE WITH ORDINARY P %

W2: "QUEEN P" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 6) & SATISFIES(D,D EQ 1)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W2) )
      & ISPAWN(A) & LOC(A,S) & RF(S,R,F) & SATISFIES(R,R EQ 7) & FILEF(S,S2)
   -> MOVECAND(D,S,S2) & STRAT:TRIED('W2,L,D) & NEGATE(1);
W2S: "Q P SUCC" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 6) & NOT SATISFIES(D,D EQ 1)
      & ISPAWN(A) & LOC(A,S) & RF(S,R,F) & SATISFIES(R,R EQ 7) & FILEF(S,S2)
      & NOT( EXISTS(A2,A3) & ISKING(A2) & NOT HASCOLOR(A2,P) & ISKING(A3)
            & VNEQ(A3,A2) & CONTROLS(A2,S2) & NOT CONTROLS(A3,S2) )
      & NOT( EXISTS(A2) & LOC(A2,S2) )
   -> SUCCSTRAT(D,P,L,'W2) & NEGATE(1);
W2Z: "Q P STAT" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 6) & NOT SATISFIES(D,D EQ 1)
      & ISPAWN(A) & LOC(A,S) & RF(S,R,F) & SATISFIES(R,R EQ 7) & FILEF(S,S2)
      & NOT( EXISTS(A2,A3) & ISKING(A2) & NOT HASCOLOR(A2,P) & ISKING(A3)
            & VNEQ(A3,A2) & CONTROLS(A2,S2) & NOT CONTROLS(A3,S2) )
      & NOT( EXISTS(A2) & LOC(A2,S2) )
   -> TERMAWIN(P,'W2Z) & NEGATE(1);

W3: "ADV P" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W3) )
      & ISPAWN(A) & LOC(A,S1) & FILEF(S1,S2) & NOT( EXISTS(A2) & LOC(A2,S2) )
      & RF(S1,R1,F1) & NOT SATISFIES(R1,R1 EQ 7)
      & ISKING(A2) & NOT HASCOLOR(A2,P) & LOC(A2,S3) & RF(S3,R2,F2)
      & SATISFIES2(R1,R2,NOT(R2 %LESS R1))
      & SATISFIES2(F1,F2,ABS(F2-F1) %LESS (9-R1))  % BK IN SQUARE %
   -> MOVECAND(D,S1,S2) & STRAT:TRIED('W3,L,D) & NEGATE(1);
W3A: "ADV P 1" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5) & SATISFIES(D,D EQ 1) & ISPAWN(A1)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W3A) )
      & HASCOLOR(A1,P) & LOC(A1,S1) & ISKING(A2) & NOT HASCOLOR(A2,P)
      & LOC(A2,S2) & RF(S1,R1,F1) & RF(S2,R2,F2)
      & NOT( SATISFIES2(R1,R2,NOT(R2 %LESS R1))       % BK OUT OF SQUARE %
            & SATISFIES2(F1,F2,ABS(F2-F1) %LESS (9-R1)))
   -> MOVECAND(D,S1,S2) & STRAT:TRIED('W3A,L,D) & NEGATE(1);
W3K: "ADV P K" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W3K) )
      & ISPAWN(A1) & LOC(A1,S1) & FILEF(S1,S2) & ISKING(A2) & LOC(A2,S2)
      & HASCOLOR(A2,P) & CONTROLS(A2,S3) & RF(S3,R3,F3) & RF(S2,R2,F2)
      & NOT SATISFIES2(R2,R3,R3 %LESS R2)
   -> MOVECAND(D,S2,S3) & STRAT:TRIED('W3K,L,D) & NEGATE(1);
W3L: "ADV P K7" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W3L) )
      & ISPAWN(A1) & LOC(A1,S1) & RF(S1,R1,F1) & SATISFIES(R1,R1 EQ 7)
      & ISKING(A2) & HASCOLOR(A2,P) & FILEF(S1,S2) & NOT CONTROLS(A2,S2)
      & LOC(A2,S2) & FILEB(S1,S3) & RANKL(S1,S4) & RANKR(S1,S5)
      & DIAGLB(S1,S6) & DIAGRB(S1,S7)
   -> MOVE:TO(D,A1,S3) & MOVE:TO(D,A1,S4) & MOVE:TO(D,A1,S5) & MOVE:TO(D,A1,S6)
      & MOVE:TO(D,A1,S7) & MEANSHOLD(D) & STRAT:TRIED('W3L,L,D) & NEGATE(1);

W3B: "ADV P SUCC" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5) & SATISFIES(D,D %GREAT 1) & ISPAWN(A1)
      & HASCOLOR(A1,P) & LOC(A1,S1) & ISKING(A2) & NOT HASCOLOR(A2,P)
      & LOC(A2,S2) & RF(S1,R1,F1) & RF(S2,R2,F2)
      & NOT( SATISFIES2(R1,R2,NOT(R2 %LESS R1))      % BK OUT OF SQUARE %
            & SATISFIES2(F1,F2,ABS(F2-F1) %LESS (9-R1)))
      & NOT( EXISTS(A3,S3) & FILEF(S1,S3) & LOC(A3,S3) )
   -> SUCCSTRAT(D,P,L,'W3) & NEGATE(1);
W3V: "ADV P OK?" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5) & ISPAWN(A1) & LOC(A1,S1) & RF(S1,R1,F1)
      & SATISFIES(R1,R1 EQ 7) & ISKING(A2) & HASCOLOR(A2,P) & CONTROLS(A2,S1)
      & ISKING(A3) & VNEQ(A3,A2) & LOC(A3,S3) & FILEF(S1,S3)
      & LOC(A2,S2) & NOT DIAGRB(S1,S2) & NOT DIAGLB(S1,S2)
   -> TERMAWIN(P,'W3V) & NEGATE(1);
W3W: "ADV P OK?" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5) & ISPAWN(A1) & LOC(A1,S1) & RF(S1,R1,F1)
      & SATISFIES(R1,R1 EQ 7) & ISKING(A2) & HASCOLOR(A2,P) & CONTROLS(A2,S1)
      & ISKING(A3) & VNEQ(A3,A2) & LOC(A3,S3) & NOT FILEF(S1,S3)
   -> TERMAWIN(P,'W3W) & NEGATE(1);
      % W3Y WOULD FAIL ON THAT IF BK CONTROLLED Q SQUARE %
W3Y: "ADV P OK" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5) & ISPAWN(A1) & HASCOLOR(A1,P) & LOC(A1,S1)
      & ISKING(A2) & HASCOLOR(A2,P) & ISKING(A3) & VNEQ(A3,A2)
      & LOC(A3,S3) & FILEF(S1,S4) & VNEQ(S3,S4)
      & NOT( CONTROLS(A3,S4) & NOT CONTROLS(A2,S4) )
      & LOC(A2,S2) & RF(S1,R1,F1) & RF(S2,R2,F2) & RF(S3,R3,F3)
      & SATISFIES2(R1,R3,NOT(R3 %LESS R1))
      & SATISFIES2(F1,F3,ABS(F3-F1) %LESS (9-R1))  % BK IN SQUARE %
      & NOT SATISFIES3(R1,R2,R3,MAX(ABS(R3-R1),ABS(F3-F1))
            %LESS MAX(ABS(R2-R1),ABS(F2-F1)))     % NOT BK CLOSER TO P %
   -> TERMAWIN(P,'W3Y) & NEGATE(1);
W3Z: "ADV P STAT" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 5) & ISPAWN(A1) & HASCOLOR(A1,P) & LOC(A1,S1)
      & ISKING(A2) & NOT HASCOLOR(A2,P) & LOC(A2,S2) & RF(S1,R1,F1)
      & RF(S2,R2,F2)
      & NOT( SATISFIES2(R1,R2,NOT(R2 %LESS R1))      % BK NOT IN SQUARE %
            & SATISFIES2(F1,F2,ABS(F2-F1) %LESS (9-R1)))
      % IF WK DIR IN FRONT OF P, W3Y IS ALSO TRUE BUT WON'T BE APPLICABLE
            IF BK OUT OF SQUARE, SO DON'T CHECK FOR THAT HERE %
   -> TERMAWIN(P,'W3Z) & NEGATE(1);

W4: "CONTR P" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 4)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W4) )
      & ISPAWN(A1) & ISKING(A2) & HASCOLOR(A1,C) & HASCOLOR(A2,C)
      & LOC(A1,S1) & FILEF(S1,S2) & FILEF(S2,S3) & NOT CONTROLS(A2,S2)
   -> MOVE:TOWARD(D,A2,S3) & STRAT:TRIED('W4,L,D) & NEGATE(1);
W4Z: "CONTR P STAT" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 4)
      & ISPAWN(A1) & ISKING(A2) & HASCOLOR(A2,P)
      & LOC(A1,S1) & FILEF(S1,S2) & FILEF(S2,S3) & CONTROLS(A2,S3)
   -> TERMAWIN(P,'W4Z) & NEGATE(1);

      % W5 - W7Y ARE USED BY B ALSO, EXCEPT W5Z, W60, AND W6P %

W5: "DEFEND P" = SELECTSTRAT(D,P) & KPKHASP(P2) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 3)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W5) )
      & ISPAWN(A1) & ISKING(A2) & HASCOLOR(A2,P) & LOC(A1,S1)
   -> MOVE:TOWARD(D,A2,S1) & STRAT:TRIED('W5,L,D) & NEGATE(1);
W5Z: "DEFEND P" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 3) & ISPAWN(A1) & ISKING(A2) & HASCOLOR(A2,P)
      & LOC(A1,S1) & CONTROLS(A2,S1)
   -> TERMAWIN(P,'W5Z) & NEGATE(1);

W6: "TOWARD K" = SELECTSTRAT(D,P) & KPKHASP(P2) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 2) & NOT SATISFIES(D,D EQ 2)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W6) )
      & ISKING(A1) & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A2,A1) & LOC(A2,S)
   -> MOVE:TOWARD(D,A1,S) & STRAT:TRIED('W6,L,D) & NEGATE(1);
W60: "OPPOS K" = SELECTSTRAT(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 2)
      & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W60) )
      & ISKING(A1) & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A1,A2)
      & LOC(A2,S) & FILEB(S,S1) & FILEB(S1,S2) & CONTROLS(A1,S2)
      % ASSUMES WK ALWAYS BEHIND BK %
   -> MOVE:TO(D,A1,S2) & STRAT:TRIED('W60,L,D) & NEGATE(1);
W6P: "OPPOS K" = SELECTSTATIC(D,P) & KPKHASP(P) & CURLEVEL(D,L)
      & SATISFIES(L,L EQ 2)
      & ISKING(A1) & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A1,A2)
      & LOC(A2,S) & FILEB(S,S1) & FILEB(S1,S2) & CONTROLS(A1,S2)
      % ASSUMES WK ALWAYS BEHIND BK %
   -> TERMAWIN(P,'W6P) & NEGATE(1);
```

```
W6W: "TOWARD K" = SELECTSTATIC(D,P) & KPKHASP(P2) & CURLEVEL(D,L)
        & SATISFIES(L,L EQ 2) & NOT SATISFIES(D,D EQ 2)
        & ISKING(A1) & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A2,A1) & LOC(A2,S)
    -> MOVE:TOWARD(D,A1,S) & MEANSHOLD(D) & MEANSEXAM(D) & W6WRESEXAM(D,A1)
        & NEGATE(1);
W6X: "TOWARD K RES" = W6WRESEXAM(D,A1) & MOVEEXAM(D,S1,S2) & HASCOLOR(A1,P)
        % NOT UNIQUE, NECESSARILY %
        & NOT( EXISTS(A2) & ISKING(A2) & NOT HASCOLOR(A2,P) & CONTROLS(A2,S2) )
    -> W6WRESEXAM(D,A1) & TERM:WIN(P,'W6X) & NEGATE(2);
W6Y: "TOWARD K RES-" = W6WRESEXAM(D,A1) & MOVEEXAM(D,S1,S2) & ISKING(A2)
        & VNEQ(A2,A1) & CONTROLS(A2,S2)
    -> W6WRESEXAM(D,A1) & NEGATE(2);
W6Z: "TOWARD K RESF" = W6WRESEXAM(D,A1)
        & NOT( EXISTS(S1,S2) & MOVEEXAM(D,S1,S2) )
    -> NEGATE(1);

W7: "ELSE" = SELECTSTRAT(D,P) & KPKHASP(P2) & CURLEVEL(D,L)
        & SATISFIES(L,L EQ 1) & NOT SATISFIES(D,D EQ 2) & ISKING(A1)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'W7) )
        & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A2,A1) & LOC(A2,S1) & ISPAWN(A3)
        & LOC(A3,S2)
    -> MOVE:TOWARD(D,A1,S1) & MOVE:TOWARD(D,A1,S2) & MEANSHOLD(D)
        & MEANSEXAM(D) & W7RESEXAM(D,A1) & STRAT:TRIED('W7,L,D) & NEGATE(1);
W7A: "ELSE RES" = W7RESEXAM(D,A1) & LOC(A1,S1) & CONTROLS(A1,S2)
        & NOT MOVEEXAM(D,S1,S2)
    -> W7RESERS(D) & MOVE:CAND(D,S1,S2) & NEGATE(1);
W7B: "ELSE RES-" = W7RESEXAM(D,A1) & LOC(A1,S1)
        & NOT( EXISTS(S2) & CONTROLS(A1,S2) & NOT MOVEEXAM(D,S1,S2) )
    -> W[illegible]ERS(D) & NEGATE(1);
W7[illegible] [illegible] = W7RESERS(D) & MOVEEXAM(D,S1,S2) -> NEGATE(1,2);
W7W: [illegible] STAT" = SELECTSTATIC(D,P) & KPKHASP(P2) & CURLEVEL(D,L)
        & SATISFIES(L,L EQ 1) & NOT SATISFIES(D,D EQ 2) & ISKING(A1)
        & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A2,A1) & LOC(A2,S1) & ISPAWN(A3)
        & LOC(A3,S2)
    -> MOVE:TOWARD(D,A1,S1) & MOVE:TOWARD(D,A1,S2) & MEANSHOLD(D)
        & MEANSEXAM(D) & W7WRESEXAM(D,A1) & NEGATE(1);
W7X: "ELSE RES STAT" = W7WRESEXAM(D,A1) & LOC(A1,S1) & CONTROLS(A1,S2)
        & NOT MOVEEXAM(D,S1,S2) & HASCOLOR(A1,P)
    -> W7RESERS(D) & TERM:WIN(P,'W7X) & NEGATE(1);
W7Y: "ELSE RES- STAT" = W7WRESEXAM(D,A1) & LOC(A1,S1)
        & NOT( EXISTS(S2) & CONTROLS(A1,S2) & NOT MOVEEXAM(D,S1,S2) )
    -> W7RESERS(D) & NEGATE(1);

    END;
```

. . . .

```
EXPR KPKB(); BEGIN     % BLACK STRATEGIES %          % PAGE 8 %

B1: "CAP P" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 7) & SATISFIES(D,D EQ 1)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'B1) )
        & ISPAWN(A1) & LOC(A1,S) & CONTROLS(A2,S) & ISKING(A2)
        & NOT HASCOLOR(A1,P) & HASCOLOR(A2,P) & LOC(A2,S2)
        & NOT( EXISTS(A3) & CONTROLS(A3,S) & NOT HASCOLOR(A3,P) )
    -> MOVE:CAND(D,S2,S) & STRAT:TRIED('B1,L,D) & NEGATE(1);
B1S: "CAP P" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 7) & NOT SATISFIES(D,D EQ 1)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'B1) )
        & ISPAWN(A1) & LOC(A1,S) & CONTROLS(A2,S) & ISKING(A2)
        & NOT HASCOLOR(A1,P) & HASCOLOR(A2,P) & LOC(A2,S2)
        & NOT( EXISTS(A3) & CONTROLS(A3,S) & NOT HASCOLOR(A3,P) )
    -> SUCCSTRAT(D,P,L,'B1S) & NEGATE(1);
B1Z: "CAP P" = SELECTSTATIC(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 7) & ISPAWN(A1) & LOC(A1,S)
        & CONTROLS(A2,S) & ISKING(A2) & NOT HASCOLOR(A1,P) & HASCOLOR(A2,P)
        & LOC(A2,S2)
        & NOT( EXISTS(A3) & CONTROLS(A3,S) & NOT HASCOLOR(A3,P) )
    -> TERM:WIN(P,'B1Z) & NEGATE(1);

B2: "UNDER P" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 6) & SATISFIES(D,D EQ 1)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'B2) )
        & ISPAWN(A1) & NOT HASCOLOR(A1,P) & LOC(A1,S1) & FILEF(S1,S2)
        & RF(S2,R1,F1) & SATISFIES(R1,R1 EQ 8) & ISKING(A2) & HASCOLOR(A2,P)
        & CONTROLS(A2,S2) & ISKING(A3) & VNEQ(A3,A2) & FILEB(S1,S3)
        & NOT LOC(A3,S3)
    -> MOVE:CAND(D,S1,S2) & STRAT:TRIED('B2,L,D) & NEGATE(1);
B2A: "UNDER P-" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 6) & SATISFIES(D,D ?GREAT 1)
        & ISPAWN(A1) & NOT HASCOLOR(A1,P) & LOC(A1,S1) & FILEF(S1,S2)
        & RF(S2,R1,F1) & SATISFIES(R1,R1 EQ 8) & ISKING(A2) & HASCOLOR(A2,P)
        & CONTROLS(A2,S2) & ISKING(A3) & VNEQ(A3,A2) & FILEB(S1,S3)
        & NOT LOC(A3,S3)
    -> SUCCSTRAT(D,P,L,'B2A) & NEGATE(1);
B2B: "UNDER P STAT" = SELECTSTATIC(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 6)
        & ISPAWN(A1) & NOT HASCOLOR(A1,P) & LOC(A1,S1) & FILEF(S1,S2)
        & RF(S2,R1,F1) & SATISFIES(R1,R1 EQ 8) & ISKING(A2) & HASCOLOR(A2,P)
        & CONTROLS(A2,S2) & ISKING(A3) & VNEQ(A3,A2) & FILEB(S1,S3)
        & NOT LOC(A3,S3)
    -> TERM:WIN(P,'B2B) & NEGATE(1);
B2Q: "Q EDGE STALE" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 6) & SATISFIES(D,D EQ 1)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'B2Q) )
        & ISPAWN(A1) & NOT HASCOLOR(A1,P) & LOC(A1,S1) & RF(S1,R1,F1)
        & SATISFIES(R1,R1 EQ 7) & SATISFIES(F1,F1 MEMQ '(1 8)) & ISKING(A2)
        & HASCOLOR(A2,P)
    -> MOVE:TO(D,A2,'A7) & MOVE:TO(D,A2,'H7) & MEANSHOLD(D)
        & STRAT:TRIED('B2Q,L,D) & NEGATE(1);

B3: "INTERC P" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 5)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'B3) )
        & ISPAWN(A1) & LOC(A1,S1) & ISKING(A2) & HASCOLOR(A2,P) & RF(S1,R1,F1)
        & LOC(A2,S2) & RF(S2,R2,F2) & SATISFIES2(R1,R2,R2 ?GREAT R1-2)
        & SATISFIES3(F1,F2,R1,ABS(F2-F1) ?LESS 10-R1)  % BK IN SQUARE %
        & RF(S3,R3,F1) & SATISFIES(R3,R3 EQ 8)
        % TOWARD QUEENING SQUARE %
    -> MOVE:TOWARD(D,A2,S3) & STRAT:TRIED('B3,L,D) & NEGATE(1);
B3Z: "INTERC P STAT" = SELECTSTATIC(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 5)
        & ISPAWN(A1) & LOC(A1,S1) & ISKING(A2) & HASCOLOR(A2,P) & RF(S1,R1,F1)
        & LOC(A2,S2) & RF(S2,R2,F2) & SATISFIES2(R1,R2,R2 ?GREAT R1-2)
        & SATISFIES3(F1,F2,R1,ABS(F2-F1) ?LESS 10-R1)  % BK IN SQUARE %
        & ISKING(A3) & NOT HASCOLOR(A3,P) & LOC(A3,S3) & RF(S3,R3,F3)
        & NOT( SATISFIES3(R1,R2,R3,R3 ?GREAT R2-1)
             % WK BETWEEN BK & WP Q SQUARE %
             & SATISFIES3(F1,F2,F3,NOT(F2 ?GREAT F3) & NOT(F3 ?GREAT F1)
                        OR NOT(F1 ?GREAT F2) & NOT(F3 ?GREAT F2)) )
        & RF(S4,R4,F1) & SATISFIES(R4,R4 EQ 8) & NOT CONTROLS(A3,S4)
    -> TERM:WIN(P,'B3Z) & NEGATE(1);

B4: "BLOCK P" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 4)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'B4) )
        & ISPAWN(A1) & ISKING(A2) & HASCOLOR(A2,P) & NOT HASCOLOR(A1,P)
        & LOC(A1,S1) & RF(S1,R1,F1) & RF(S2,R2,F1)
        & SATISFIES2(R2,R1,R2 ?GREAT R1)
    -> MOVE:TOWARD(D,A2,S2) & MEANSHOLD(D) & STRAT:TRIED('B4,L,D) & NEGATE(1);
B4Z: "BLOCK P STAT" = SELECTSTATIC(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 4)
        & ISPAWN(A1) & ISKING(A2) & HASCOLOR(A2,P) & LOC(A1,S1)
        & RF(S1,R1,F1) & LOC(A2,S2) & RF(S2,R2,F1)
        & SATISFIES2(R1,R2,R2 ?GREAT R1)
    -> TERM:WIN(P,'B4Z) & NEGATE(1);

%    B5 THROUGH B7 ARE SAME AS W5 THROUGH W7, EXCEPT AS FOLLOWS %

B5Z: "ATTACK P" = SELECTSTATIC(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 3) & ISPAWN(A1) & ISKING(A2)
        & HASCOLOR(A2,P) & LOC(A1,S1) & CONTROLS(A2,S1)
        & NOT( EXISTS(A3) & ISKING(A3) & VNEQ(A3,A2) & CONTROLS(A3,S1) )
    -> TERM:WIN(P,'B5Z) & NEGATE(1);

B6O: "OPPOS K" = SELECTSTRAT(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 2)
        & NOT( EXISTS(X) & STRAT:TRIED(X,L,D) & SATISFIES(X,X EQ 'B6O) )
        & ISKING(A1) & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A1,A2)
        & LOC(A2,S) & FILEF(S,S1) & FILEF(S1,S2) & CONTROLS(A1,S2)
        % ASSUMES WK ALWAYS BEHIND BK %
    -> MOVE:TO(D,A1,S2) & STRAT:TRIED('B6O,L,D) & NEGATE(1);
B6P: "OPPOS K" = SELECTSTATIC(D,P) & NOT KPKHASP(P) & KPKHASP(C)
        & CURLEVEL(D,L) & SATISFIES(L,L EQ 2)
        & ISKING(A1) & HASCOLOR(A1,P) & ISKING(A2) & VNEQ(A1,A2)
        & LOC(A2,S) & FILEF(S,S1) & FILEF(S1,S2) & CONTROLS(A1,S2)
        % ASSUMES WK ALWAYS BEHIND BK %
    -> TERM:WIN(P,'B6P) & NEGATE(1);

    END;
```

. . . .

```
EXPR KPKX(); BEGIN          % EXAMPLES FOR TESTING %  % PAGE 7 %

X1; "TEST 1" = TEST1(X)
   -> CONTROLSK('WK) & CONTROLSK('BK) & CONTROLSP('WP) & LOC('WK,'E4)
      & LOC('BK,'C7) & LOC('WP,'E6) & ISKING('BK) & ISKING('WK) & ISPAWN('WP)
      & HASCOLOR('BK,'B) & HASCOLOR('WK,'W) & HASCOLOR('WP,'W) & KPKINIT(X);

X1K; "CONTROLS FOR K" = CONTROLSK(A) & LOC(A,S) & FILEF(S,S1)
      & FILEB(S,S2) & DIAGLF(S,S3) & DIAGRF(S,S4) & RANKL(S,S5) & RANKR(S,S6)
      & DIAGLB(S,S7) & DIAGRB(S,S8)
   -> CONTROLS(A,S1) & CONTROLS(A,S2) & CONTROLS(A,S3) & CONTROLS(A,S4)
      & CONTROLS(A,S5) & CONTROLS(A,S6) & CONTROLS(A,S7) & CONTROLS(A,S8)
      & NEGATE(1);
X1P; "CONTROLS P" = CONTROLSP(A) & LOC(A,S) & DIAGRF(S,S1) & DIAGLF(S,S2)
   -> CONTROLS(A,S1) & CONTROLS(A,S2) & NEGATE(1);

X2; "TEST 2" = TEST2(X)
   -> CONTROLSK('WK) & CONTROLSK('BK) & CONTROLSP('WP) & LOC('WK,'E6)
      & LOC('BK,'E8) & LOC('WP,'E5) & ISKING('BK) & ISKING('WK) & ISPAWN('WP)
      & HASCOLOR('BK,'B) & HASCOLOR('WK,'W) & HASCOLOR('WP,'W) & KPKINIT(X);

X3; "TEST 3" = TEST3(X)
   -> CONTROLSK('WK) & CONTROLSK('BK) & CONTROLSP('WP) & LOC('WK,'E6)
      & LOC('BK,'E8) & LOC('WP,'E4) & ISKING('BK) & ISKING('WK) & ISPAWN('WP)
      & HASCOLOR('BK,'B) & HASCOLOR('WK,'W) & HASCOLOR('WP,'W) & KPKINIT(X);

   END;
END.
```

Appendix B. CROSS-REFERENCE OF PREDICATES

```
ADDPROOF
 RHSUSES S84 -S84 S85 -S85 S86 -S86
ASCEND
 LHSUSES S7
 RHSUSES -S7 S13 S25
CAPTURED
 LHSUSES S8
 NESTEDL S9
 RHSUSES -S8 Q8
CHANGELEVEL
 LHSUSES S15 S16 S17 S18
 RHSUSES S4 -S15 -S16 -S17 -S18
CHECKCAP
 LHSUSES Q8 Q9
 RHSUSES -Q8 -Q9 Q11
CHECKMOVERESULT
 LHSUSES S23 S25
 NESTEDL S21
 RHSUSES S21 S21A -S23 -S25
CHECKOTHERSTRAT
 LHSUSES S3 S4
 NESTEDL S23
 RHSUSES S3 -S4 -S7 S15 S17 S38 S42 S43
CHECKTERM
 LHSUSES S30 S31 S32 S33 S34 S35 S36 S36O S36R S37L S37R
 RHSUSES S1 S5 S6 -S7 S30 -S31 -S32 -S33 -S34 -S35 -S36 -S36O -S36R -S37L
 -S37R -S38 -S39
CONTROLLED
 LHSUSES S8C
 RHSUSES -S8C Q8C
CONTROLS
 LHSUSES S34 S37L S37R Q3 Q4 Q7 Q8C Q11 Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 -M1
 -M2 -M3 -M4 -M5 -M6 -M7 -M8 M9 M16 -M17 W3K -W3L W3V W3W -W6 W6Z W5Z W8O W8P
 W8Y W7A W7X B1 B1S B1Z B2 B2A B2B -B3Z B5Z B6O B6P
 NESTEDL S31 -S31 S33 S34 Q11 W2S -W2S W2Z -W2Z W3Y -W3Y W6X W7B W7Y B1 B1S
 B1Z B5Z
 RHSUSES S8C Q3 -Q3 Q4 -Q4 -Q8C Q11 -Q11 Q12 -Q12 Q13 -Q13 Q14 -Q14 Q15 -Q15
 Q16 -Q16 Q17 -Q17 Q18 -Q18 Q19 -Q19 X1K X1P
CONTROLSK
 LHSUSES X1K
 RHSUSES X1 -X1K X2 X3
CONTROLSP
 LHSUSES X1P
 RHSUSES X1 -X1P X2 X3
CURLEVEL
 LHSUSES S5 S6 S7 S15 S16 S17 S18 S42 S43 S60 S61 S62 Q0 Q0C W2 W2S W2Z W3 W3A
  W3K W3L W3S W3V W3W W3Y W3Z W4 W4Z W5 W5Z W6 W6O W6P W6W W7 W7W B1 B1S B1Z
 B2 B2A B2B B2Q B3 B3Z B4 B4Z B5Z B6O B6P
 RHSUSES S1 S5 S6 -S7 S15 -S15 S17 -S17 S42 -S42 S43 -S43
DEPTH
 LHSUSES S3 S4 S5 S6 S7 S15 S16 S17 S18 S21 S21A S21D S210 S23 S25 S41 S42 S43
  S50 S51 Q0 Q0C Q1 Q2 Q7 Q8
 RHSUSES S1 S5 -S5 S6 -S6 S7 -S7 S16 S18 S210 S210
DESCEND
 LHSUSES S5 S6
 RHSUSES -S5 -S6 S21 S21A
DIAGLB
 LHSUSES Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 M2 M3 M5 W3L -W3V X1K
DIAGLF
 LHSUSES Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 M5 M7 M8 X1K X1P
DIAGRB
 LHSUSES Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 M1 M2 M4 W3L -W3V X1K
DIAGRF
 LHSUSES Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19 M4 M6 M7 X1K X1P
ERSCHECKTERM
 LHSUSES S38 S39
 RHSUSES S1 S5 S6 -S7 -S31 -S32 -S33 -S34 -S35 -S36 -S36O -S36R -S37L -S37R
 -S38 -S39
ERSMOVES
 LHSUSES S26 S26N
 RHSUSES S23 -S26 -S26N
ERSSTRATTRIED
 LHSUSES S7E S7F
 RHSUSES S4 S7 -S7E -S7F
FILEB
 LHSUSES S35 Q4 Q12 Q13 Q16 Q17 Q18 Q19 M1 M2 M3 W3L W8O W8P B2 B2A B2B X1K
FILEF
 LHSUSES S36O Q3 Q12 Q13 Q16 Q17 Q18 Q19 M6 M7 M8 W2 W2S W2Z W3 W3K W3L W3V
 -W3W W3Y W4 W4Z B2 B2A B2B B6O B6P X1K
 NESTEDL W3S
```

```
FINDMOVE
  LHSUSES S1
  RHSUSES -S1
HASCOLOR
  LHSUSES S0 S11 S210 S31 S33 S34 S35 S36 S360 S36R S37L S37R S60 S61 S62 Q7
  -W3 W3A -W3A W3K W3L W3S -W3S W3V W3W W3Y W3Z -W3Z W4 W4Z W5 W5Z W6 W60 W6P
  W6W W6X W7 W7W W7X B1 -B1 B1S -B1S B1Z -B1Z B2 -B2 B2A -B2A B2B -B2B B2Q -B2Q
   B3 B3Z -B3Z B4 -B4 B4Z B5Z B60 B6P
  NESTEDL S21 S33 S34 Q3 Q11 -W2S -W2Z -W6X -B1 -B1S -B1Z
  RHSUSES X1 X2 X3
ISKING
  LHSUSES S31 S33 S34 S35 S36 S360 S36R S37L S37R S50 S51 S60 S61 S62 Q7 Q11 W3
   W3A W3K W3L W3S W3V W3W W3Y W3Z W4 W4Z W5 W5Z W6 W60 W6P W6W W6Y W7 W7W B1
  B1S B1Z B2 B2A B2B B2Q B3 B3Z B4 B4Z B5Z B60 B6P
  NESTEDL W2S W2Z W6X B5Z
  RHSUSES X1 X2 X3
ISPAWN
  LHSUSES S0 S21 S31 S35 S36 S360 S36R S37L S37R S50 S60 S61 S62 Q3 Q4 W2 W2S
  W2Z W3 W3A W3K W3L W3S W3V W3W W3Y W3Z W4 W4Z W5 W5Z W7 W7W B1 B1S B1Z B2 B2A
   B2B B2Q B3 B3Z B4 B4Z B5Z
  NESTEDL S32 S51
  RHSUSES X1 X2 X3
KPKHASP
  LHSUSES S31 S32 -S32 S33 -S33 S35 S36 S360 -S360 S36R S37L -S37L S37R -S37R
  S50 S51 S60 S61 S62 W2 W2S W2Z W3 W3A W3K W3L W3S W3V W3W W3Y W3Z W4 W4Z W5
  W5Z W6 W60 W6P W6W W7 W7W B1 -B1 B1S -B1S B1Z -B1Z B2 -B2 B2A -B2A B2B -B2B
  B2Q -B2Q B3 -B3 B3Z -B3Z B4 -B4 B4Z -B4Z B5Z -B5Z B60 -B60 B6P -B6P
  RHSUSES S0
KPKINIT
  LHSUSES S0
  RHSUSES X1 X2 X3
LASTPN
  LHSUSES S64 S65 S66
  RHSUSES S0 S64 -S64 S65 -S65 S66 -S66
LOC
  LHSUSES S11 S21 S21D S31 S33 S34 S35 S36 S360 S36R S37L S37R S50 S51 S60 S61
  S62 Q0 Q0C Q3 Q4 Q7 Q8 Q11 M1 M2 M3 M4 M5 M6 M7 M8 M9 M9N M16 W2 W2S W2Z W3
  W3A W3K W3L W3S W3V W3W W3Y W3Z W4 W4Z W5 W5Z W6 W60 W6P W6W W7 W7A W7B W7W
  W7X W7Y B1 B1S B1Z B2 -B2 B2A -B2A B2B -B2B B2Q B3 B3Z B4 B4Z B5Z B60 B6P X1K
   X1P
  NESTEDL S21 S32 S33 S34 S51 Q0 Q3 Q8 Q11 W2S W2Z W3 W3S
  RHSUSES S8 Q3 -Q3 Q4 -Q4 -Q8 Q11 -Q11 X1 X2 X3
MAKEMOVE
  LHSUSES Q3 Q4 Q7 Q11
  RHSUSES Q0 Q0C Q1 -Q3 -Q4 -Q7 -Q11
MAKEMOVEK
  LHSUSES Q12 Q13 Q14 Q15 Q16 Q17 Q18 Q19
  RHSUSES Q11 -Q12 -Q13 -Q14 -Q15 -Q16 -Q17 -Q18 -Q19
MAKEMOVE:T
  LHSUSES Q0 Q0C
  RHSUSES S5 S6 -Q0 -Q0C
MAXDEPTH
  LHSUSES S30
  RHSUSES S0
MAXSLEVEL
  LHSUSES S1 S17 S18 S42
  RHSUSES S0
MEANSEXAM
  LHSUSES -M13 M13X
  RHSUSES -M13X W6W W7 W7W
MEANSHOLD
  LHSUSES -M11 M12
  RHSUSES -M12 W3L W6W W7 W7W B2Q B4
MEANSRELS
  LHSUSES M13 M13X M14
  RHSUSES M12 -M13 -M13X -M14
MINSLEVEL
  LHSUSES S15 S16 S43
  RHSUSES S0
MOVECAND
  LHSUSES S21 S21A S21D S21D S26
  NESTEDL S3 S21 S23 S26N
  RHSUSES -S21 -S21A -S21D -S21D -S26 M11 M13 W2 W3 W3A W3K W7A B1 B2
MOVEEXAM
  LHSUSES W6X W6Y -W7A W7C -W7X
  NESTEDL W6Z -W7B -W7Y
  RHSUSES M13X -W6X -W6Y -W7C
MOVEHIST
  LHSUSES S5 S6 S7 S50 S51
  RHSUSES S1 S5 S6 S7 -S7
MOVEHOLD
  LHSUSES M11 M13 M13X
  NESTEDL M14
  RHSUSES M1 M2 M3 M4 M5 M6 M7 M8 M9 -M11 -M13 -M13X M16
MOVE:TO
  LHSUSES M16 M17
  RHSUSES -M16 -M17 W3L W60 B2Q B60
MOVE:TOWARD
  LHSUSES M1 M2 M3 M4 M5 M6 M7 M8 M9 M9Y M9N
  RHSUSES -M1 -M2 -M3 -M4 -M5 -M6 -M7 -M8 -M9 -M9Y -M9N W4 W5 W6 W6W W7 W7W B7
  B4
MOVER
  LHSUSES S5 S6 S7 S15 S16 S17 S18 S41 S42 S43
  RHSUSES S1 S5 -S5 S6 -S6 S7 -S7
MOVING
  RHSUSES S11
NODECOUNT
  LHSUSES Q0 Q0C
  RHSUSES S1 Q0 -Q0 Q0C -Q0C
OFFBOARD
  LHSUSES S210 -Q3 -Q11 -M9 M9Y
  NESTEDL -S33 -S34
PLAYER
  LHSUSES S5 S6 S7 S32
  RHSUSES S0
PRINTBOARD
  LHSUSES S50 S51 S52
  RHSUSES S1 S23 S24 S41 S42 -S50 -S51 -S52
PRINTEDBOARD
  LHSUSES -S50 -S51 S52
  RHSUSES S50 S51 -Q0 -Q0C
RANKL
  LHSUSES Q14 Q15 Q16 Q17 Q18 Q19 M3 M5 M8 W3L X1K
RANKR
  LHSUSES Q14 Q15 Q16 Q17 Q18 Q19 M1 M4 M6 W3L X1K
RECORDOLD
  LHSUSES S64 S65 S66 S68
  RHSUSES S63 -S68
RECORDDONE
  LHSUSES S68
  NESTEDL S64 S65 S66
  RHSUSES S64 S65 S66 -S68
RECORDFIN
  LHSUSES S67
  RHSUSES S63 -S67
RECORDFIN2
  LHSUSES S68
  RHSUSES S67 -S68
RECORDPRE
  LHSUSES S63
  RHSUSES S60 S61 S62 -S63
RECORDWIN
  LHSUSES S60 S61 S62
  RHSUSES S23 -S60 -S61 -S62
REFUTED
  LHSUSES -S23 S25
  NESTEDL S3
  RHSUSES S13 S24 -S25 S41 Q7
RESTORECAP
  LHSUSES S8 S9
  RHSUSES S7 -S8 -S9
RESTORECON
  LHSUSES S8C
  RHSUSES S8 -S8C
RETRACTHOLD
  LHSUSES -Q1 Q2
  RHSUSES -Q2 Q7
RETRACTMOVE
  LHSUSES Q1 Q2
  RHSUSES S7 -Q1 -Q2
RETRACTING
  LHSUSES -Q7
  NESTEDL -Q11
  RHSUSES Q1 -Q4 -Q11
SF
  LHSUSES S21 S31 S35 S36 S36R S50 S51 M1 M2 M3 M4 M5 M6 M7 M8 W2 W2S W2Z W3
  W3A W3K W3L W3S W3V W3W W3Y W3Z B2 B2A B2B B2Q B3 B3Z B4 B4Z
  NESTEDL S21
SAVECON
  LHSUSES Q8C
  RHSUSES Q8 -Q8C
SELECTSTATIC
  LHSUSES W2Z W3V W3W W3Y W3Z W4Z W5Z W6P W6W W7W B1Z B2B B3Z B4Z B5Z B6P
  RHSUSES S39 -W2Z -W3V -W3W -W3Y -W3Z -W4Z -W5Z -W6P -W6W -W7W -B1Z -B2B -B3Z
  -B4Z -B5Z -B6P
SELECTSTRAT
```

```
  LHSUSES -S3 W2 W2S W3 W3A W3K W3L W3B W4 W5 W6 W60 W7 B1 B1S B2 B2A B2Q B3 B4
   B60
  NESTEDL -S4
  RHSUSES S3 -S4 -S7 S15 S17 S38 -W2 -W2S -W3 -W3A -W3K -W3L -W3B -W4 -W5 -W6
  -W60 -W7 -B1 -B1S -B2 -B2A -B2Q -B3 -B4 -B60
STATICEVAL
  LHSUSES -S30 S39
  RHSUSES S30 -S39 -S41 -S42 -S43
STRATiTRIED
  LHSUSES S7E
  NESTEDL S7F W2 W3 W3A W3K W3L W4 W5 W6 W60 W7 B1 B1S B2 B2Q B3 B4 B60
  RHSUSES -S7E W2 W3 W3A W3K W3L W4 W5 W6 W60 W7 B1 B2 B2Q B3 B4 B60
SUCCISTRAT
  LHSUSES S24
  RHSUSES -S24 W2S W3S B1S B2A
SUCCEED
  LHSUSES S11 S13
  NESTEDL S3
  RHSUSES -S11 -S13 S23
TERMAWIN
  LHSUSES S41 S42 S43
  RHSUSES S31 S32 S33 S34 S35 S36 S360 S36R S37L S37R -S41 -S42 -S43 W2Z W3V
  W3W W3Y W3Z W4Z W5Z W6P W6X W7X B1Z B2B B3Z B4Z B5Z B6P
TEST1
  LHSUSES X1
TEST2
  LHSUSES X2
TEST3
  LHSUSES X3
TRACING
  RHSUSES S15 S16 S17 S18 S21A S23 S24 S41 S42 S43 S50 S51 S64 Q0 Q0C Q1 Q2
W6WRESEXAM
  LHSUSES W6X W6Y W6Z
  RHSUSES W6W W6X W6Y -W6Z
W7RESERS
  LHSUSES W7C
  RHSUSES W7A W7B -W7C W7X W7Y
W7RESEXAM
  LHSUSES W7A W7B
  RHSUSES W7 -W7A -W7B
W7WRESEXAM
  LHSUSES W7X W7Y
  RHSUSES W7W -W7X -W7Y
WINCAND
  LHSUSES S21A
  NESTEDL S21
  RHSUSES -S21A
```

Appendix C. <u>SAMPLE CREATED [illegible] PRODUCTIONS</u>

```
LISTING OF KPKPW CREATED BY 40 NODE RUN

PV-1
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE B))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E7))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE E4))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE C7))) (SATISFIES D (EQ D 1))
  (CURLEVEL D L) (SATISFIES L (EQ L 5))
RHS (WINCAND D (QUOTE C7) (QUOTE D7))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
PN-1
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE B))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E7))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE E4))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE C7)))
  (SATISFIES D (AND (NEQ D 1) (NOT (*LESS D 2)))) (CURLEVEL D L)
  (SATISFIES L (NOT (*GREAT L 5)))
RHS (TERMAWIN P (QUOTE PN-1)) (NOT (CHECK:TERM D P)) (NOT (ERSCHECK:TERM D P))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
PN-2
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE W))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E6))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE D6))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE E8)))
  (SATISFIES D (AND (NEQ D 1) (NOT (*LESS D 5)))) (CURLEVEL D L)
  (SATISFIES L (NOT (*GREAT L 5)))
RHS (TERMAWIN P (QUOTE PN-2)) (NOT (CHECK:TERM D P)) (NOT (ERSCHECK:TERM D P))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
PW-1
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE W))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E6))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE D6))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE E8))) (SATISFIES D (*LESS D 8))
  (CURLEVEL D L) (SATISFIES L (EQ L 5))
RHS (WINCAND D (QUOTE E6) (QUOTE E7))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
PV-2
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE W))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E6))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE D6))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE E8))) (SATISFIES D (EQ D 1))
  (CURLEVEL D L) (SATISFIES L (EQ L 5))
RHS (WINCAND D (QUOTE E6) (QUOTE E7))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
PW-2
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE W))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E6))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE E5))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE D8))) (SATISFIES D (*LESS D 3))
  (CURLEVEL D L) (SATISFIES L (EQ L 4))
RHS (WINCAND D (QUOTE E5) (QUOTE D6))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
        .
        .
        .
PV-9
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE W))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E6))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE E4))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE C7))) (SATISFIES D (EQ D 1))
  (CURLEVEL D L) (SATISFIES L (EQ L 4))
RHS (WINCAND D (QUOTE E4) (QUOTE E5))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
PN-9
LHS (CHECK:TERM D P) (SATISFIES P (EQ P (QUOTE W))) (ISPAWN A1) (LOC A1 S1)
  (SATISFIES S1 (EQ S1 (QUOTE E6))) (ISKING A2) (LOC A2 S2)
  (SATISFIES S2 (EQ S2 (QUOTE E4))) (HASCOLOR A2 P2) (KPKHASP P2) (ISKING A3)
  (LOC A3 S3) (SATISFIES S3 (EQ S3 (QUOTE C7)))
  (SATISFIES D (AND (NEQ D 1) (NOT (*LESS D 1)))) (CURLEVEL D L)
  (SATISFIES L (NOT (*GREAT L 4)))
RHS (TERMAWIN P (QUOTE PN-9)) (NOT (CHECK:TERM D P)) (NOT (ERSCHECK:TERM D P))
VARS L S3 A3 P2 S2 A2 S1 A1 P D
```

Appendix D. DETAILED BEHAVIOR ON TEST1

TEST1: ORDINARY VERSION WITH P BUILDING

```
  ..  ..  ..  ..
..  BK  ..  ..
  ..  ..WP..  ..
..  ..  ..  ..
  ..  ..WK..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 5 W
1 MOVING WP FROM E6 TO E7 LEVEL 5
. 2 MOVING BK FROM C7 TO D8 LEVEL 5
  CAN'T MOVE C7 D8
. 3 MOVING BK FROM C7 TO D7 LEVEL 5
    LEVEL + 6 W
    LEVEL + FAIL DEPTH 3 W
    SUCCEED C7 D7 = 523
      ..  ..  ..  ..
    ..  ..BKWP  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..WK..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (E6 E7) (C7 D7)
    ADDPROD PN-1 DEPTH 2 LEVEL 5 C7 D7
  RETRACTING C7 D7
RETRACTING E6 E7
LEVEL - 4 W
4 MOVING WK FROM E4 TO E5 LEVEL 4
. 5 MOVING BK FROM C7 TO D8 LEVEL 4
. . 6 MOVING WK FROM E5 TO D6 LEVEL 4
. . . 7 MOVING BK FROM D8 TO E8 LEVEL 4
. . . . 8 MOVING WK FROM D6 TO E7 LEVEL 4
        CAN'T MOVE D6 E7
. . . . 9 MOVING WK FROM D6 TO D7 LEVEL 4
        CAN'T MOVE D6 D7
        LEVEL + 5 W
. . . . 10 MOVING WP FROM E6 TO E7 LEVEL 4
          LEVEL + 5 B
          LEVEL + 6 B
          LEVEL + 7 B
          LEVEL + FAIL DEPTH 6 B
          SUCCEED E6 E7 = 523
            ..  ..BK..  ..
          ..  ..  WP  ..
            ..  WK  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 D8) (E5 D6) (D8 E8) (E6 E7)
          ADDPROD PN-2 DEPTH 5 LEVEL 5 E6 E7
        RETRACTING E6 E7
      RETRACTING D8 E8
. . . 11 MOVING BK FROM D8 TO E7 LEVEL 4
      CAN'T MOVE D8 E7
      LEVEL + 5 B
. . . 12 MOVING BK FROM D8 TO E8 LEVEL 4
        TERMINAL WIN FOR W = PN-2
          ..  ..BK..  ..
        ..  ..  ..  ..
          ..  WKWP..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 D8) (E5 D6) (D8 E8)
      RETRACTING D8 E8
      LEVEL + 6 B
      LEVEL + 7 B
      LEVEL + FAIL DEPTH 4 B
      SUCCEED E5 D6 = 523
      ADDPROD PN-3 DEPTH 3 LEVEL 4 E5 D6
    RETRACTING E5 D6
  RETRACTING C7 D8
. 13 MOVING BK FROM C7 TO D7 LEVEL 4
  CAN'T MOVE C7 D7
. 14 MOVING BK FROM C7 TO C8 LEVEL 4
. . 15 MOVING WK FROM E5 TO D6 LEVEL 4
. . . 16 MOVING BK FROM C8 TO D8 LEVEL 4
        TERMINAL WIN FOR B = 536
          ..  BK  ..  ..
        ..  ..  ..  ..
          ..  WKWP..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 C8) (E5 D6) (C8 D8)
        LEVEL + FAIL DEPTH 5 W
        SUCCEED C8 D8 = 523
        ADDPROD PN-4 DEPTH 4 LEVEL 4 C8 D8
      RETRACTING C8 D8
    RETRACTING E5 D6
. . 17 MOVING WK FROM E5 TO F6 LEVEL 4
. . . 18 MOVING BK FROM C8 TO D8 LEVEL 4
. . . . 19 MOVING WK FROM F6 TO E7 LEVEL 4
        CAN'T MOVE F6 E7
. . . . 20 MOVING WK FROM F6 TO F7 LEVEL 4
. . . . . 21 MOVING BK FROM D8 TO E8 LEVEL 4
          CAN'T MOVE D8 E8
. . . . . 22 MOVING BK FROM D8 TO E7 LEVEL 4
          CAN'T MOVE D8 E7
          LEVEL + 5 B
. . . . . 23 MOVING BK FROM D8 TO E8 LEVEL 4
          CAN'T MOVE D8 E8
          LEVEL + 6 B
          LEVEL + 7 B
          LEVEL + FAIL DEPTH 6 B
          SUCCEED F6 F7 = 523
            ..  BK  ..  ..
          ..  ..  ..WK..
            ..  ..WP..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 C8) (E5 F6) (C8 D8) (F6 F7)
          ADDPROD PN-5 DEPTH 5 LEVEL 4 F6 F7
        RETRACTING F6 F7
      RETRACTING C8 D8
. . . 24 MOVING BK FROM C8 TO D7 LEVEL 4
      CAN'T MOVE C8 D7
. . . 25 MOVING BK FROM C8 TO C7 LEVEL 4
. . . . 26 MOVING WK FROM F6 TO E7 LEVEL 4
. . . . . 27 MOVING BK FROM C7 TO D8 LEVEL 4
          CAN'T MOVE C7 D8
. . . . . 28 MOVING BK FROM C7 TO D7 LEVEL 4
          CAN'T MOVE C7 D7
. . . . . 29 MOVING BK FROM C7 TO C8 LEVEL 4
            LEVEL + 5 W
. . . . . . 30 MOVING WK FROM E7 TO E8 LEVEL 4
              TERMINAL WIN FOR W = 536
                ..BK..WK..  ..
              ..  ..  ..  ..
                ..  ..WP..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
                ..  ......  ..
              ..  ..  ..  ..
              (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8) (E7 E8)
              LEVEL + FAIL DEPTH 8 B
              SUCCEED E7 E8 = 523
              ADDPROD PN-6 DEPTH 7 LEVEL 5 E7 E8
            RETRACTING E7 E8
          RETRACTING C7 C8
. . . . . 31 MOVING BK FROM C7 TO D6 LEVEL 4
          CAN'T MOVE C7 D6
          LEVEL + 5 B
. . . . . 32 MOVING BK FROM C7 TO D8 LEVEL 4
          CAN'T MOVE C7 D8
. . . . . 33 MOVING BK FROM C7 TO D7 LEVEL 4
          CAN'T MOVE C7 D7
. . . . . 34 MOVING BK FROM C7 TO C8 LEVEL 4
            TERMINAL WIN FOR W = PN-6
              ..BK..  ..  ..
```

D.

```
             .. .. WK ..
               .. ..WP.. ..
             .. .. .. ..
               .. .. .. ..
             .. .. .. ..
               .. .. .. ..
             .. .. .. ..
             (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8)
           RETRACTING C7 C8
           LEVEL + 6 B
           LEVEL + 7 B
           LEVEL + FAIL DEPTH 6 B
           SUCCEED F6 E7 = S23
           ADDPROD PN-7 DEPTH 5 LEVEL 4 F6 E7
         RETRACTING F6 E7
       RETRACTING C8 C7
       LEVEL + 5 B
 . . . 36 MOVING BK FROM C8 TO D8 LEVEL 4
         TERMINAL WIN FOR W = PN-5
           .. BK .. ..
         .. .. .. ..
           .. ..WPWK ..
         .. .. .. ..
           .. .. .. ..
         .. .. .. ..
           .. .. .. ..
         .. .. .. ..
         (E4 E5) (C7 C8) (E5 F8) (C8 D8)
       RETRACTING C8 D8
 . . . 36 MOVING BK FROM C8 TO D7 LEVEL 4
       CAN'T MOVE C8 D7
       LEVEL + 6 B
       LEVEL + 7 B
       LEVEL + FAIL DEPTH 4 B
       SUCCEED E5 F6 = S23
         ..BK.. .. ..
       .. .. .. ..
         .. ..WPWK ..
       .. .. .. ..
         .. .. .. ..
       .. .. .. ..
         .. .. .. ..
       .. .. .. ..
       (E4 E5) (C7 C8) (E5 F6)
       ADDPROD PN-8 DEPTH 3 LEVEL 4 E5 F6
     RETRACTING E5 F6
   RETRACTING C7 C8
 . 37 MOVING BK FROM C7 TO D6 LEVEL 4
   CAN'T MOVE C7 D6
   LEVEL + 5 B
 . 38 MOVING BK FROM C7 TO D8 LEVEL 4
     TERMINAL WIN FOR W = PN-3
       .. BK .. ..
     .. .. .. ..
       .. ..WP.. ..
     .. .. WK ..
       .. .. .. ..
     .. .. .. ..
       .. .. .. ..
     .. .. .. ..
     (E4 E5) (C7 D8)
   RETRACTING C7 D8
 . 39 MOVING BK FROM C7 TO D7 LEVEL 4
   CAN'T MOVE C7 D7
 . 40 MOVING BK FROM C7 TO C8 LEVEL 4
     TERMINAL WIN FOR W = PN-8
       ..BK.. .. ..
     .. .. .. ..
       .. ..WP.. ..
     .. .. WK ..
       .. .. .. ..
     .. .. .. ..
       .. .. .. ..
     .. .. .. ..
     (E4 E5) (C7 C8)
   RETRACTING C7 C8
   LEVEL + 6 B
   LEVEL + 7 B
   LEVEL + FAIL DEPTH 2 B
   SUCCEED E4 E5 = S23
   ADDPROD PN-9 DEPTH 1 LEVEL 4 E4 E5
MOVING (W WK E4 E5)
```

```
RUN TIME 14 MIN. 4.22 SEC

EXAM    TRY    FIRE   WMACT  E/F    E/T    T/F
8839    2688   842    3737   8.12   2.54   3.18
0.123   0.314  1.00   0.226  SEC AVG

1906 INSERTS 1832 DELETES 486 WARNINGS 25 NEW OBJECTS
MAX :SMPX LENGTH 136
CORE (FREE.FULL): (5881 . 2538) USED (8887 . 458)

:ACTS LOADPS (KPKEG . EXP) (KPKEGM . MAC) (KPKO . EXP) (KPKM . EXP) (KPKW . EXP)
  (KPKB . EXP) (KPKX . EXP) KPKC RESTOREDB (KPKEGB . DBS) SAVEPS (CLOSED (
  KPKEG . EXP)) RUN SMPXEMPTY SAVEDB (CLOSED (KPK1D1 . DBS)) SAVEPS (CLOSED (
  KPK1D1 . EXP)) (CLOSED (KPK1D1 . TRS)) SMPXEMPTY

FIRED 58 OUT OF 166 PRODS
CHECK:OTHER:STRAT (1 W)
CONTROLS (BK D7) (BK D8) (BK B8) (BK B7) (BK C8) (BK B6) (BK C6) (BK D6) (WK E6)
  (WK F5) (WK F6) (WK F4) (WK D4) (WK D6) (WK E4) (WK D5) (WP D7) (WP F7)
CUR:LEVEL (1 4) (2 7)
DEPTH (2)
DIAGLB (A1 @0) (A2 @1) (A3 @2) (A4 @3) (A5 @4) (A6 @5) (A7 @6) (A8 @7) (B1 A0)
  (B2 A1) (B3 A2) (B4 A3) (B5 A4) (B6 A5) (B7 A6) (B8 A7) (B9 A8) (C1 B0)
  (C2 B1) (C3 B2) (C4 B3) (C5 B4) (C6 B5) (C7 B6) (C8 B7) (C9 B8) (D1 C0)
  (D2 C1) (D3 C2) (D4 C3) (D5 C4) (D6 C5) (D7 C6) (D8 C7) (D9 C8) (E1 D0)
  (E2 D1) (E3 D2) (E4 D3) (E5 D4) (E6 D5) (E7 D6) (E8 D7) (E9 D8) (F1 E0)
  (F2 E1) (F3 E2) (F4 E3) (F5 E4) (F6 E5) (F7 E6) (F8 E7) (F9 E8) (G1 F0)
  (G2 F1) (G3 F2) (G4 F3) (G5 F4) (G6 F5) (G7 F6) (G8 F7) (G9 F8) (H1 G0)
  (H2 G1) (H3 G2) (H4 G3) (H5 G4) (H6 G5) (H7 G6) (H8 G7) (H9 G8) (I2 H1)
  (I3 H2) (I4 H3) (I5 H4) (I6 H5) (I7 H6) (I8 H7) (I9 H8)
DIAGLF (A1 @2) (A2 @3) (A3 @4) (A4 @5) (A5 @6) (A6 @7) (A7 @8) (A8 @9) (B0 A1)
  (B1 A2) (B2 A3) (B3 A4) (B4 A5) (B5 A6) (B6 A7) (B7 A8) (B8 A9) (C0 B1)
  (C1 B2) (C2 B3) (C3 B4) (C4 B5) (C5 B6) (C6 B7) (C7 B8) (C8 B9) (D0 C1)
  (D1 C2) (D2 C3) (D3 C4) (D4 C5) (D5 C6) (D6 C7) (D7 C8) (D8 C9) (E0 D1)
  (E1 D2) (E2 D3) (E3 D4) (E4 D5) (E5 D6) (E6 D7) (E7 D8) (E8 D9) (F0 E1)
  (F1 E2) (F2 E3) (F3 E4) (F4 E5) (F5 E6) (F6 E7) (F7 E8) (F8 E9) (G0 F1)
  (G1 F2) (G2 F3) (G3 F4) (G4 F5) (G5 F6) (G6 F7) (G7 F8) (G8 F9) (H0 G1)
  (H1 G2) (H2 G3) (H3 G4) (H4 G5) (H5 G6) (H6 G7) (H7 G8) (H8 G9) (I0 H1)
  (I1 H2) (I2 H3) (I3 H4) (I4 H5) (I5 H6) (I6 H7) (I7 H8)
DIAGRB (@2 A1) (@3 A2) (@4 A3) (@5 A4) (@6 A5) (@7 A6) (@8 A7) (@9 A8) (A1 B0)
  (A2 B1) (A3 B2) (A4 B3) (A5 B4) (A6 B5) (A7 B6) (A8 B7) (A9 B8) (B1 C0)
  (B2 C1) (B3 C2) (B4 C3) (B5 C4) (B6 C5) (B7 C6) (B8 C7) (B9 C8) (C1 D0)
  (C2 D1) (C3 D2) (C4 D3) (C5 D4) (C6 D5) (C7 D6) (C8 D7) (C9 D8) (D1 E0)
  (D2 E1) (D3 E2) (D4 E3) (D5 E4) (D6 E5) (D7 E6) (D8 E7) (D9 E8) (E1 F0)
  (E2 F1) (E3 F2) (E4 F3) (E5 F4) (E6 F5) (E7 F6) (E8 F7) (E9 F8) (F1 G0)
  (F2 G1) (F3 G2) (F4 G3) (F5 G4) (F6 G5) (F7 G6) (F8 G7) (F9 G8) (G1 H0)
  (G2 H1) (G3 H2) (G4 H3) (G5 H4) (G6 H5) (G7 H6) (G8 H7) (G9 H8) (H1 I0)
  (H2 I1) (H3 I2) (H4 I3) (H5 I4) (H6 I5) (H7 I6) (H8 I7)
DIAGRF (@0 A1) (@1 A2) (@2 A3) (@3 A4) (@4 A5) (@5 A6) (@6 A7) (@7 A8) (A0 B1)
  (A1 B2) (A2 B3) (A3 B4) (A4 B5) (A5 B6) (A6 B7) (A7 B8) (A8 B9) (B0 C1)
  (B1 C2) (B2 C3) (B3 C4) (B4 C5) (B5 C6) (B6 C7) (B7 C8) (B8 C9) (C0 D1)
  (C1 D2) (C2 D3) (C3 D4) (C4 D5) (C5 D6) (C6 D7) (C7 D8) (C8 D9) (D0 E1)
  (D1 E2) (D2 E3) (D3 E4) (D4 E5) (D5 E6) (D6 E7) (D7 E8) (D8 E9) (E0 F1)
  (E1 F2) (E2 F3) (E3 F4) (E4 F5) (E5 F6) (E6 F7) (E7 F8) (E8 F9) (F0 G1)
  (F1 G2) (F2 G3) (F3 G4) (F4 G5) (F5 G6) (F6 G7) (F7 G8) (F8 G9) (G0 H1)
  (G1 H2) (G2 H3) (G3 H4) (G4 H5) (G5 H6) (G6 H7) (G7 H8) (G8 H9) (H1 I2)
  (H2 I3) (H3 I4) (H4 I5) (H5 I6) (H6 I7) (H7 I8) (H8 I9)
FILEB (A1 A0) (A2 A1) (A3 A2) (A4 A3) (A5 A4) (A6 A5) (A7 A6) (A8 A7) (A9 A8)
  (B1 B0) (B2 B1) (B3 B2) (B4 B3) (B5 B4) (B6 B5) (B7 B6) (B8 B7) (B9 B8)
  (C1 C0) (C2 C1) (C3 C2) (C4 C3) (C5 C4) (C6 C5) (C7 C6) (C8 C7) (C9 C8)
  (D1 D0) (D2 D1) (D3 D2) (D4 D3) (D5 D4) (D6 D5) (D7 D6) (D8 D7) (D9 D8)
  (E1 E0) (E2 E1) (E3 E2) (E4 E3) (E5 E4) (E6 E5) (E7 E6) (E8 E7) (E9 E8)
  (F1 F0) (F2 F1) (F3 F2) (F4 F3) (F5 F4) (F6 F5) (F7 F6) (F8 F7) (F9 F8)
  (G1 G0) (G2 G1) (G3 G2) (G4 G3) (G5 G4) (G6 G5) (G7 G6) (G8 G7) (G9 G8)
  (H1 H0) (H2 H1) (H3 H2) (H4 H3) (H5 H4) (H6 H5) (H7 H6) (H8 H7) (H9 H8)
FILEF (A0 A1) (A1 A2) (A2 A3) (A3 A4) (A4 A5) (A5 A6) (A6 A7) (A7 A8) (A8 A9)
  (B0 B1) (B1 B2) (B2 B3) (B3 B4) (B4 B5) (B5 B6) (B6 B7) (B7 B8) (B8 B9)
  (C0 C1) (C1 C2) (C2 C3) (C3 C4) (C4 C5) (C5 C6) (C6 C7) (C7 C8) (C8 C9)
  (D0 D1) (D1 D2) (D2 D3) (D3 D4) (D4 D5) (D5 D6) (D6 D7) (D7 D8) (D8 D9)
  (E0 E1) (E1 E2) (E2 E3) (E3 E4) (E4 E5) (E5 E6) (E6 E7) (E7 E8) (E8 E9)
  (F0 F1) (F1 F2) (F2 F3) (F3 F4) (F4 F5) (F5 F6) (F6 F7) (F7 F8) (F8 F9)
  (G0 G1) (G1 G2) (G2 G3) (G3 G4) (G4 G5) (G5 G6) (G6 G7) (G7 G8) (G8 G9)
  (H0 H1) (H1 H2) (H2 H3) (H3 H4) (H4 H5) (H5 H6) (H6 H7) (H7 H8) (H8 H9)
HASCOLOR (BK B) (WK W) (WP W)
ISKING (BK) (WK)
ISPAWN (WP)
KPK:HASP (W)
KPKINIT (T)
LAST:PN (PN-9)
LOC (BK C7) (WK E5) (WP E6)
MAXDEPTH (9)
MAXSLEVEL (B 7) (W 6)
MINSLEVEL (B 1) (W 1)
```

```
MOVE:HIST ((T (E4 E5)))
MOVER (B)
MOVING (W WK E4 E5)
NODE:COUNT (41)
OFFBOARD (@0) (@1) (@2) (@3) (@4) (@5) (@6) (@7) (@8) (@9) (A0) (A9) (B0) (B9)
   (C0) (C9) (D0) (D9) (E0) (E9) (F0) (F9) (G0) (G9) (H0) (H9) (I0) (I1) (I2)
   (I3) (I4) (I5) (I6) (I7) (I8) (I9)
PLAYER (B) (W)
PRINTED:BOARD (T)
RANKL (A1 @1) (A2 @2) (A3 @3) (A4 @4) (A5 @5) (A6 @6) (A7 @7) (A8 @8) (B1 A1)
   (B2 A2) (B3 A3) (B4 A4) (B5 A5) (B6 A6) (B7 A7) (B8 A8) (C1 B1) (C2 B2)
   (C3 B3) (C4 B4) (C5 B5) (C6 B6) (C7 B7) (C8 B8) (D1 C1) (D2 C2) (D3 C3)
   (D4 C4) (D5 C5) (D6 C6) (D7 C7) (D8 C8) (E1 D1) (E2 D2) (E3 D3) (E4 D4)
   (E5 D5) (E6 D6) (E7 D7) (E8 D8) (F1 E1) (F2 E2) (F3 E3) (F4 E4) (F5 E5)
   (F6 E6) (F7 E7) (F8 E8) (G1 F1) (G2 F2) (G3 F3) (G4 F4) (G5 F5) (G6 F6)
   (G7 F7) (G8 F8) (H1 G1) (H2 G2) (H3 G3) (H4 G4) (H5 G5) (H6 G6) (H7 G7)
   (H8 G8) (I1 H1) (I2 H2) (I3 H3) (I4 H4) (I5 H5) (I6 H6) (I7 H7) (I8 H8)
RANKR (@1 A1) (@2 A2) (@3 A3) (@4 A4) (@5 A5) (@6 A6) (@7 A7) (@8 A8) (A1 B1)
   (A2 B2) (A3 B3) (A4 B4) (A5 B5) (A6 B6) (A7 B7) (A8 B8) (B1 C1) (B2 C2)
   (B3 C3) (B4 C4) (B5 C5) (B6 C6) (B7 C7) (B8 C8) (C1 D1) (C2 D2) (C3 D3)
   (C4 D4) (C5 D5) (C6 D6) (C7 D7) (C8 D8) (D1 E1) (D2 E2) (D3 E3) (D4 E4)
   (D5 E5) (D6 E6) (D7 E7) (D8 E8) (E1 F1) (E2 F2) (E3 F3) (E4 F4) (E5 F5)
   (E6 F6) (E7 F7) (E8 F8) (F1 G1) (F2 G2) (F3 G3) (F4 G4) (F5 G5) (F6 G6)
   (F7 G7) (F8 G8) (G1 H1) (G2 H2) (G3 H3) (G4 H4) (G5 H5) (G6 H6) (G7 H7)
   (G8 H8) (H1 I1) (H2 I2) (H3 I3) (H4 I4) (H5 I5) (H6 I6) (H7 I7) (H8 I8)
RF (A1 1 1) (A2 2 1) (A3 3 1) (A4 4 1) (A5 5 1) (A6 6 1) (A7 7 1) (A8 8 1)
   (B1 1 2) (B2 2 2) (B3 3 2) (B4 4 2) (B5 5 2) (B6 6 2) (B7 7 2) (B8 8 2)
   (C1 1 3) (C2 2 3) (C3 3 3) (C4 4 3) (C5 5 3) (C6 6 3) (C7 7 3) (C8 8 3)
   (D1 1 4) (D2 2 4) (D3 3 4) (D4 4 4) (D5 5 4) (D6 6 4) (D7 7 4) (D8 8 4)
   (E1 1 5) (E2 2 5) (E3 3 5) (E4 4 5) (E5 5 5) (E6 6 5) (E7 7 5) (E8 8 5)
   (F1 1 6) (F2 2 6) (F3 3 6) (F4 4 6) (F5 5 6) (F6 6 6) (F7 7 6) (F8 8 6)
   (G1 1 7) (G2 2 7) (G3 3 7) (G4 4 7) (G5 5 7) (G6 6 7) (G7 7 7) (G8 8 7)
   (H1 1 8) (H2 2 8) (H3 3 8) (H4 4 8) (H5 5 8) (H6 6 8) (H7 7 8) (H8 8 8)
STRAT:TRIED (W4 4 1)
TEST1 (T)
TRACING (T)
```

- - - - - - - - - - - - - - - - - - -

```
(TEST1: ORDINARY VERSION WITH P BUILDING)
(X1-1 X1K-1 X1K-2 X1P-1 S0-1 S1-1 S50-1 S30-1 S4-1 S7F-1 S15-1 W3-1 S21-1 S6-1

        O0-1 O3-1 S30-2 B3-1 M6-1 M11-1 M11-2 M11-3 S21-2 S6-1
        O0-2 O7-1 S25-1 S7-1 S7F-2 O2-1 S9-1 S21-3 S6-2
        O0-3 O11-1 O9-1 O15-1 S30-3 S4-2 S7F-3 S17-1 S4-3 S7F-4 S10-1 S23-1
  S50-2 S20-1 S62-1 S63-1 S65-1 S64-1 S67-1 S60-1 S60-2 S13-1 S7-2 S7F-5 O1-1
  O11-2 O9-2 O14-1 S9-2 S25-2 S7-3 S7E-1 O1-2 W4-1 S9-3 S3-1 S4-4 S7E-2 S15-2
  W4-1 M7-1 M11-4 M11-5 M11-6 S21-4 S5-2
        O0-4 O11-3 O9-3 O12-1 S30-4 B4-1 B4-2 M4-1 M6-2 M12-1 M13-1 M13-2
  M13-3 M13-4 S21-5 S6-3
        O0-5 O11-4 O9-4 O17-1 S30-5 W4-2 M7-2 M11-7 M11-8 M11-9 S210-1 S21-6
  S6-4
        O0-6 O11-5 O9-5 O16-1 S30-6 B4-3 B4-4 M9-1 M8-2 M12-2 M13-5 M13-6
  S21-7 S6-5
        O0-7 O11-6 O9-6 O15-2 S30-7 W4-3 M6-3 M11-10 M11-11 M11-12 S210-2
  S21-8 S6-6
        O0-8 O7-2 S25-3 S7-4 S7F-6 O2-2 S9-4 S21-9 S6-7
        O0-9 O7-3 S25-4 S7-5 S7F-7 O2-3 S9-5 S3-2 S4-5 S7E-3 S17-2 W3-2 S21-10
  S6-8
        O0-10 O3-2 S30-8 B4-5 M8N-1 M12-3 M14-1 S3-3 S4-6 S7E-4 S17-3 B3-2
  M8N-2 S3-4 S4-7 S7E-5 S17-4 S4-8 S7F-8 S17-5 S4-9 S7F-9 S10-2 S23-2 S50-3
  S26N-1 S60-1 S63-2 S64-2 S66-1 S65-2 S67-2 S60-3 S60-4 S60-5 S13-2 S7-6
  S7F-10 O1-3 O4-2 S9-6 [illegible]-5 S7-7 S7E-6 O1-4 O11-7 O9-7 O14-2 S21-11 S6-9
        O0-11 O7-4 S25-[illegible]-8 S7F-11 O2-4 S9-7 S3-5 S4-10 S7E-7 S17-6 B3-3
  M9-3 M11-13 S21-12 S6-10
        O0-12 O11-8 O9-8 O15-3 PN-2-1 S41-1 S50-4 S25-7 S7-9 S7F-12 O1-5 O11-9
  O9-9 O14-3 S9-8 S3-6 S4-11 S7E-8 S17-7 S4-12 S7F-13 S17-8 S4-13 S7F-14 S10-3
  S23-3 S52-1 S26-2 S61-1 S63-3 S65-3 S66-2 S64-3 S67-3 S60-6 S60-7 S60-8 S13-3
  S7-10 S7F-15 O1-6 O11-10 O9-10 O19-1 S9-9 S25-8 S7-11 S7E-9 O1-7 O11-11 O9-11
  O18-1 S21-13 S6-11
        O0-13 O7-5 S25-9 S7-12 S7F-16 O2-5 S9-10 S21-14 S6-12
        O0-14 O11-12 O9-12 O12-2 S30-9 W4-4 M7-3 M11-14 M11-15 M11-16 S210-3
  S21-15 S6-13
        O0-15 O11-13 O9-13 O16-2 S30-10 B4-6 B4-7 M4-2 M1-1 M12-4 M13-7 M13-8
  M13-9 M13-10 S210-1 S21-16 S6-14
        O0-16 O11-14 O9-14 O15-4 S35-1 S42-1 S50-5 S3-7 S4-14 S7F-17 S10-4
  S23-4 S52-2 S26-3 S26-4 S62-2 S63-4 S64-4 S66-3 S65-4 S67-4 S60-9 S60-10
  S60-11 S13-4 S7-13 S7F-18 O1-8 O11-15 O9-15 O14-4 S9-11 S25-10 S7-14 S7E-10
  O1-9 O11-16 O9-16 O19-2 S21-17 S6-15
        O0-17 O11-17 O9-17 O17-2 S30-11 B4-8 B4-9 M1-2 M4-3 M12-5 M13-11
  M13-12 M13-13 M13-14 S21-18 S6-16
        O0-18 O11-18 O9-18 O15-5 S30-12 W4-5 M8-1 M11-17 M11-18 M11-19 S210-4
  S21-19 S6-17
        O0-19 O7-6 S25-11 S7-15 S7F-19 O2-6 S9-12 S9-13 S21-20 S6-18
        O0-20 O11-19 O9-19 O12-3 S30-13 B4-10 B4-11 M9-4 M9-5 M12-6 M13-15
  M13-16 S21-21 S6-19
        O0-21 O7-7 S25-12 S7-16 S7F-20 O2-7 S9-14 S21-22 S6-20
        O0-22 O7-8 S25-13 S7-17 S7F-21 O2-8 S9-15 S3-8 S4-15 S7E-11 S17-9 B3-4
  M9-6 M11-20 S21-23 S6-21
        O0-23 O7-9 S25-14 S7-18 S7F-22 O2-9 S9-16 S3-9 S4-16 S7E-12 S17-10
  S4-17 S7F-23 S17-11 S4-18 S7F-24 S10-5 S23-5 S50-6 S26N-2 S61-2 S63-5 S65-5
  S66-4 S64-5 S67-5 S60-12 S60-13 S60-14 S13-5 S7-19 S7F-25 O1-10 O11-20 O9-20
  O13-1 S9-17 S25-15 S7-20 S7E-13 O1-11 O11-21 O9-21 O14-5 S21-24 S6-22
        O0-24 O7-10 S25-16 S7-21 S7F-26 O2-10 S9-18 S210-2 S21-25 S6-23
        O0-25 O11-22 O9-22 O13-2 S30-14 W4-6 M8-2 M11-21 M11-22 M11-23 S210-5
  S21-26 S6-24
        O0-26 O11-23 O9-23 O16-3 S30-15 B4-12 B4-13 M6-4 M4-4 M12-7 M13-17
  M13-18 M13-19 M13-20 S21-27 S6-25
        O0-27 O7-11 S25-17 S7-22 S7F-27 O2-11 S9-19 S21-28 S6-26
        O0-28 O7-12 O7-13 S25-18 S7-23 S7F-28 O2-12 S9-20 S21-29 S6-27
        O0-29 O11-24 O9-24 O12-4 S30-16 S4-19 S7F-29 S17-12 W3K-1 W3K-2 W3K-3
  W3K-4 W3K-5 S21-30 S6-28
        O0-30 O11-25 O9-25 O12-5 S36-1 S42-2 S50-7 S3-10 S4-20 S7F-30 S10-6
  S23-6 S52-3 S26-5 S26-6 S28-7 S26-8 S61-3 S63-6 S64-6 S66-5 S65-6 S67-6
  S60-15 S60-16 S60-17 S13-6 S7-24 S7F-31 O1-12 O11-26 O9-26 O13-3 S9-21 S25-19
  S7-25 S7E-14 O1-13 O11-27 O9-27 O13-4 S21-31 S6-29
        O0-31 O7-14 S25-20 S7-26 S7F-32 O2-13 S9-22 S3-11 S4-21 S7E-15 S17-13
  B3-5 M6-5 M11-24 M11-25 M11-26 S21-32 S6-30
        O0-32 O7-15 S25-21 S7-27 S7F-33 O2-14 S9-23 S21-33 S6-31
        O0-33 O7-16 O7-17 S25-22 S7-28 S7F-34 O2-15 S9-24 S21-34 S6-32
        O0-34 O11-28 O9-28 O12-6 PN-6-1 S41-2 S50-8 S25-23 S7-29 S7F-35 O1-14
  O11-29 O9-29 O13-5 S9-25 S3-12 S4-22 S7E-16 S17-14 S4-23 S7F-36 S17-15 S4-24
  S7F-37 S10-7 S23-7 S52-4 S26-9 S61-4 S63-7 S65-7 S66-6 S64-7 S67-7 S60-18
  S60-19 S60-20 S13-7 S7-30 S7F-38 O1-15 O11-30 O9-30 O19-3 S9-26 S25-24 S7-31
  S7E-17 O1-16 O11-31 O9-31 O12-7 S9-27 S3-13 S4-25 S7E-18 S17-16 B3-6 M4-5
  M11-27 M11-28 M11-29 S210-3 S21-35 S6-33
        O0-35 O11-32 O9-32 O15-6 PN-5-1 S41-3 S50-9 S25-25 S7-32 S7F-39 O1-17
  O11-33 O9-33 O14-6 S21-36 S6-34
        O0-36 O7-18 S25-26 S7-33 S7F-40 O2-16 S9-28 S3-14 S4-26 S7E-19 S17-17
  S4-27 S7F-41 S17-18 S4-28 S7F-42 S10-8 S23-8 S50-10 S26N-3 S61-5 S63-8 S64-8
  S66-7 S65-8 S67-8 S60-21 S60-22 S60-23 S13-8 S7-34 S7F-43 O1-18 O11-34 O9-34
  O18-2 S9-29 S25-27 S7-35 S7E-20 O1-19 O11-35 O9-35 O13-6 S21-37 S6-35
        O0-37 O7-19 S25-28 S7-36 S7F-44 O2-17 S9-30 S3-15 S4-29 S7E-21 S17-19
  B3-7 M6-6 M11-30 M11-31 M11-32 S21-38 S6-36
        O0-38 O11-36 O9-36 O17-3 PN-3-1 S41-4 S50-11 S25-29 S7-37 S7F-45 O1-20
  O11-37 O9-37 O18-3 S21-39 S6-37
        O0-39 O7-20 S25-30 S7-38 S7F-46 O2-18 S9-31 S21-40 S6-38
        O0-40 O11-38 O9-38 O12-8 PN-8-1 S41-5 S50-12 S25-31 S7-39 S7F-47 O1-21
  O11-39 O9-39 O13-7 S9-32 S3-16 S4-30 S7E-22 S17-20 S4-31 S7F-48 S17-21 S4-32
  S7F-49 S10-9 S23-9 S52-5 S26-10 S28-11 S61-6 S63-9 S65-9 S64-9 S67-9 S60-24
  S60-25 S11-1) )
```

- - - - - - - - - - - - - - - - - - -

```
(TEST1: ORDINARY VERSION WITH P BUILDING)

                 X S O W B M P

        X1-1     X                  4....
        S0-1       S                7.......
        W3-1             W          1.
        S21-1      S                2..
        O0-1         O              2..
        S30-2      S                1.
        B3-1               B        1.
        M6-1                 M      4....
        S21-2      S                2..
        O0-2         O              2..
        S25-1      S                3...
        O2-1         O              1.
        S9-1       S                3...
        O0-3         O              4....
        S30-3      S                20....................
        O1-1         O              4....
        S9-2       S                4....
        O1-2         O              2..
        S9-3       S                5.....
        W4-1             W          1.
        M7-1                 M      4....
        S21-4      S                2..
```

```
MOVE:HIST ((T (E4 E5)))
MOVER (B)
MOVING (W WK E4 E5)
NODE:COUNT (41)
OFFBOARD (e0) (e1) (e2) (e3) (e4) (e5) (e6) (e7) (e8) (e9) (A0) (A9) (B0) (B9)
  (C0) (C9) (D0) (D9) (E0) (E9) (F0) (F9) (G0) (G9) (H0) (H9) (I0) (I1) (I2)
  (I3) (I4) (I5) (I6) (I7) (I8) (I9)
PLAYER (B) (W)
PRINTED:BOARD (T)
RANKL (A1 e1) (A2 e2) (A3 e3) (A4 e4) (A5 e5) (A6 e6) (A7 e7) (A8 e8) (B1 A1)
  (B2 A2) (B3 A3) (B4 A4) (B5 A5) (B6 A6) (B7 A7) (B8 A8) (C1 B1) (C2 B2)
  (C3 B3) (C4 B4) (C5 B5) (C6 B6) (C7 B7) (C8 B8) (D1 C1) (D2 C2) (D3 C3)
  (D4 C4) (D5 C5) (D6 C6) (D7 C7) (D8 C8) (E1 D1) (E2 D2) (E3 D3) (E4 D4)
  (E5 D5) (E6 D6) (E7 D7) (E8 D8) (F1 E1) (F2 E2) (F3 E3) (F4 E4) (F5 E5)
  (F6 E6) (F7 E7) (F8 E8) (G1 F1) (G2 F2) (G3 F3) (G4 F4) (G5 F5) (G6 F6)
  (G7 F7) (G8 F8) (H1 G1) (H2 G2) (H3 G3) (H4 G4) (H5 G5) (H6 G6) (H7 G7)
  (H8 G8) (I1 H1) (I2 H2) (I3 H3) (I4 H4) (I5 H5) (I6 H6) (I7 H7) (I8 H8)
RANKR (e1 A1) (e2 A2) (e3 A3) (e4 A4) (e5 A5) (e6 A6) (e7 A7) (e8 A8) (A1 B1)
  (A2 B2) (A3 B3) (A4 B4) (A5 B5) (A6 B6) (A7 B7) (A8 B8) (B1 C1) (B2 C2)
  (B3 C3) (B4 C4) (B5 C5) (B6 C6) (B7 C7) (B8 C8) (C1 D1) (C2 D2) (C3 D3)
  (C4 D4) (C5 D5) (C6 D6) (C7 D7) (C8 D8) (D1 E1) (D2 E2) (D3 E3) (D4 E4)
  (D5 E5) (D6 E6) (D7 E7) (D8 E8) (E1 F1) (E2 F2) (E3 F3) (E4 F4) (E5 F5)
  (E6 F6) (E7 F7) (E8 F8) (F1 G1) (F2 G2) (F3 G3) (F4 G4) (F5 G5) (F6 G6)
  (F7 G7) (F8 G8) (G1 H1) (G2 H2) (G3 H3) (G4 H4) (G5 H5) (G6 H6) (G7 H7)
  (G8 H8) (H1 I1) (H2 I2) (H3 I3) (H4 I4) (H5 I5) (H6 I6) (H7 I7) (H8 I8)
RF (A1 1 1) (A2 2 1) (A3 3 1) (A4 4 1) (A5 5 1) (A6 6 1) (A7 7 1) (A8 8 1)
  (B1 1 2) (B2 2 2) (B3 3 2) (B4 4 2) (B5 5 2) (B6 6 2) (B7 7 2) (B8 8 2)
  (C1 1 3) (C2 2 3) (C3 3 3) (C4 4 3) (C5 5 3) (C6 6 3) (C7 7 3) (C8 8 3)
  (D1 1 4) (D2 2 4) (D3 3 4) (D4 4 4) (D5 5 4) (D6 6 4) (D7 7 4) (D8 8 4)
  (E1 1 5) (E2 2 5) (E3 3 5) (E4 4 5) (E5 5 5) (E6 6 5) (E7 7 5) (E8 8 5)
  (F1 1 6) (F2 2 6) (F3 3 6) (F4 4 6) (F5 5 6) (F6 6 6) (F7 7 6) (F8 8 6)
  (G1 1 7) (G2 2 7) (G3 3 7) (G4 4 7) (G5 5 7) (G6 6 7) (G7 7 7) (G8 8 7)
  (H1 1 8) (H2 2 8) (H3 3 8) (H4 4 8) (H5 5 8) (H6 6 8) (H7 7 8) (H8 8 8)
STRAT:TRIED (W4 4 1)
TEST1 (T)
TRACING (T)
```

- - - - - - - - - - - - - - - - - - - -

```
(TEST1: ORDINARY VERSION WITH P BUILDING)
(X1-1 X1K-1 X1K-2 X1P-1 S0-1 S1-1 S50-1 S30-1 S4-1 S7F-1 S15-1 W3-1 S21-1 S6-1

        O0-1 O3-1 S30-2 B3-1 M6-1 M11-1 M11-2 M11-3 S21-2 S6-1
        O0-2 O7-1 S25-1 S7-1 S7F-2 O2-1 S9-1 S21-3 S6-2
        O0-3 O11-1 O9-1 O15-1 S30-3 S4-2 S7F-3 S17-1 S4-3 S7F-4 S18-1 S23-1
 S50-2 S20-1 S62-1 S63-1 S65-1 S64-1 S67-1 S60-1 S60-2 S13-1 S7-2 S7F-5 O1-1
 O11-2 O9-2 O14-1 S9-2 S25-2 S7-3 S7E-1 O1-2 W4-1 S9-3 S3-1 S4-4 S7E-2 S15-2
 W4-1 M7-1 M11-4 M11-5 M11-6 S21-4 S5-2
        O0-4 O11-3 O9-3 O12-1 S30-4 B4-1 B4-2 W4-1 M6-2 M12-1 M13-1 M13-2
 M13-3 M13-4 S21-5 S6-3
        O0-5 O11-4 O9-4 O17-1 S30-5 W4-2 M7-2 M11-7 M11-8 M11-9 S210-1 S21-6
 S6-4
        O0-6 O11-5 O9-5 O16-1 S30-6 B4-3 B4-4 M9-1 M9-2 M12-2 M13-5 M13-6
 S21-7 S6-5
        O0-7 O11-6 O9-6 O15-2 S30-7 W4-3 M6-3 M11-10 M11-11 M11-12 S210-2
 S21-8 S6-6
        O0-8 O7-2 S25-3 S7-4 S7F-6 O2-2 S9-4 S21-9 S6-7
        O0-9 O7-3 S25-4 S7-5 S7F-7 O2-3 S9-5 S3-2 S4-5 S7E-3 S17-2 W3-2 S21-10
 S6-8
        O0-10 O3-2 S30-8 B4-5 M9N-1 M12-3 M14-1 S3-3 S4-6 S7E-4 S17-3 B3-2
 M9N-2 S3-4 S4-7 S7E-5 S17-4 S4-8 S7F-8 S17-5 S4-9 S7F-9 S18-2 S23-2 S50-3
 S26N-1 S60-1 S63-2 S64-2 S66-1 S65-2 S67-2 S60-3 S60-4 S60-5 S13-2 S7-6
 S7F-10 O1-3 O4-2 S9-6 [illegible]-5 S7-7 S7E-6 O1-4 O11-7 O9-7 O14-2 S21-11 S6-9
        O0-11 O7-4 S25-[illegible]-8 S7F-11 O2-4 S9-7 S3-5 S4-10 S7E-7 S17-6 B3-3
 M9-3 M11-13 S21-12 S6-10
        O0-12 O11-8 O9-8 O15-3 PN-2-1 S41-1 S50-4 S25-7 S7-9 S7F-12 O1-5 O11-9
 O9-9 O14-3 S9-8 S3-6 S4-11 S7E-8 S17-7 S4-12 S7F-13 S17-8 S4-13 S7F-14 S18-3
 S23-3 S52-1 S26-2 S61-1 S63-3 S65-3 S66-2 S64-3 S67-3 S60-6 S60-7 S60-8 S13-3
 S7-10 S7F-15 O1-6 O11-10 O9-10 O19-1 S9-9 S25-8 S7-11 S7E-9 O1-7 O11-11 O9-11
 O18-1 S21-13 S6-11
        O0-13 O7-5 S25-9 S7-12 S7F-16 O2-5 S9-10 S21-14 S6-12
        O0-14 O11-12 O9-12 O12-2 S30-9 W4-4 M7-3 M11-14 M11-15 M11-16 S210-3
 S21-15 S6-13
        O0-15 O11-13 O9-13 O16-2 S30-10 B4-6 B4-7 W4-2 M1-1 M12-4 M13-7 M13-8
 M13-9 M13-10 S210-1 S21-16 S6-14
        O0-16 O11-14 O9-14 O15-4 S35-1 S42-1 S50-5 S3-7 S4-14 S7F-17 S18-4
 S23-4 S52-2 S26-3 S26-4 S62-2 S63-4 S64-4 S66-3 S65-4 S67-4 S60-9 S60-10
 S60-11 S13-4 S7-13 S7F-18 O1-8 O11-15 O9-15 O14-4 S9-11 S25-10 S7-14 S7E-10
 O1-9 O11-16 O9-16 O19-2 S21-17 S6-15
        O0-17 O11-17 O9-17 O17-2 S30-11 B4-8 B4-9 M1-2 W4-3 M12-5 M13-11
 M13-12 M13-13 M13-14 S21-18 S6-16
        O0-18 O11-18 O9-18 O15-5 S30-12 W4-5 M8-1 M11-17 M11-18 M11-19 S210-4
 S21-19 S6-17
        O0-19 O7-6 S25-11 S7-15 S7F-19 O2-6 S9-12 S9-13 S21-20 S6-18
        O0-20 O11-19 O9-19 O12-3 S30-13 B4-10 B4-11 M9-4 M9-5 M12-6 M13-15
 M13-16 S21-21 S6-19
        O0-21 O7-7 S25-12 S7-16 S7F-20 O2-7 S9-14 S21-22 S6-20
        O0-22 O7-8 S25-13 S7-17 S7F-21 O2-8 S9-15 S3-8 S4-15 S7E-11 S17-9 B3-4
 M9-6 M11-20 S21-23 S6-21
        O0-23 O7-9 S25-14 S7-18 S7F-22 O2-9 S9-16 S3-9 S4-16 S7E-12 S17-10
 S4-17 S7F-23 S17-11 S4-18 S7F-24 S18-5 S23-5 S60-8 S26N-2 S61-2 S63-5 S65-5
 S66-4 S64-5 S67-5 S60-12 S60-13 S60-14 S13-5 S7-19 S7F-25 O1-10 O11-20 O9-20
 O13-1 S9-17 S25-15 S7-20 S7E-13 O1-11 O11-21 O9-21 O14-5 S21-24 S6-22
        O0-24 O7-10 S25-16 S7-21 S7F-26 O2-10 S9-18 S210-2 S21-25 S6-23
        O0-25 O11-22 O9-22 O13-2 S30-14 W4-6 M8-2 M11-21 M11-22 M11-23 S210-5
 S21-26 S6-24
        O0-26 O11-23 O9-23 O16-3 S30-15 B4-12 B4-13 M6-4 W4-4 M12-7 M13-17
 M13-18 M13-19 M13-20 S21-27 S6-25
        O0-27 O7-11 S25-17 S7-22 S7F-27 O2-11 S9-19 S21-28 S6-26
        O0-28 O7-12 O7-13 S25-18 S7-23 S7F-28 O2-12 S9-20 S21-29 S6-27
        O0-29 O11-24 O9-24 O12-4 S30-16 S4-19 S7F-29 S17-12 W3K-1 W3K-2 W3K-3
 W3K-4 W3K-5 S21-30 S6-28
        O0-30 O11-25 O9-25 O12-5 S36-1 S42-2 S50-7 S3-10 S4-20 S7F-30 S18-6
 S23-6 S52-3 S26-5 S26-6 S26-7 S26-8 S61-3 S63-6 S64-6 S66-5 S65-6 S67-6
 S60-15 S60-16 S60-17 S13-6 S7-24 S7F-31 O1-12 O11-26 O9-26 O13-3 S9-21 S25-19
 S7-25 S7E-14 O1-13 O11-27 O9-27 O13-4 S21-31 S6-29
        O0-31 O7-14 S25-20 S7-26 S7F-32 O2-13 S9-22 S3-11 S4-21 S7E-15 S17-13
 B3-5 M6-5 M11-24 M11-25 M11-26 S21-32 S6-30
        O0-32 O7-15 S25-21 S7-27 S7F-33 O2-14 S9-23 S21-33 S6-31
        O0-33 O7-16 O7-17 S25-22 S7-28 S7F-34 O2-15 S9-24 S21-34 S6-32
        O0-34 O11-28 O9-28 O12-6 PN-6-1 S41-2 S50-8 S25-23 S7-29 S7F-35 O1-14
 O11-29 O9-29 O13-5 S9-25 S3-12 S4-22 S7E-16 S17-14 S4-23 S7F-36 S17-15 S4-24
 S7F-37 S18-7 S23-7 S52-4 S26-9 S61-4 S63-7 S65-7 S66-6 S64-7 S67-7 S60-18
 S60-19 S60-20 S13-7 S7-30 S7F-38 O1-15 O11-30 O9-30 O19-3 S9-26 S25-24 S7-31
 S7E-17 O1-16 O11-31 O9-31 O12-7 S9-27 S3-13 S4-25 S7E-18 S17-16 B3-6 W4-5
 M11-27 M11-28 M11-29 S210-3 S21-35 S6-33
        O0-35 O11-32 O9-32 O15-6 PN-5-1 S41-3 S50-9 S25-25 S7-32 S7F-39 O1-17
 O11-33 O9-33 O14-6 S21-36 S6-34
        O0-36 O7-18 S25-26 S7-33 S7F-40 O2-16 S9-28 S3-14 S4-26 S7E-19 S17-17
 S4-27 S7F-41 S17-18 S4-28 S7F-42 S18-8 S23-8 S50-10 S26N-3 S61-5 S63-8 S64-8
 S66-7 S65-8 S67-8 S60-21 S60-22 S60-23 S13-8 S7-34 S7F-43 O1-18 O11-34 O9-34
 O18-2 S9-29 S25-27 S7-35 S7E-20 O1-19 O11-35 O9-35 O13-6 S21-37 S6-35
        O0-37 O7-19 S25-28 S7-36 S7F-44 O2-17 S9-30 S3-15 S4-28 S7E-21 S17-19
 B3-7 M6-6 M11-30 M11-31 M11-32 S21-38 S6-36
        O0-38 O11-36 O9-36 O17-3 PN-3-1 S41-4 S50-11 S25-29 S7-37 S7F-45 O1-20
 O11-37 O9-37 O18-3 S21-39 S6-37
        O0-39 O7-20 S25-30 S7-38 S7F-46 O2-18 S9-31 S21-40 S6-38
        O0-40 O11-38 O9-38 O12-8 PN-8-1 S41-5 S50-12 S25-31 S7-39 S7F-47 O1-21
 O11-39 O9-39 O13-7 S9-32 S3-16 S4-30 S7E-22 S17-20 S4-31 S7F-48 S17-21 S4-32
 S7F-49 S18-9 S23-9 S52-5 S26-10 S26-11 S61-6 S63-9 S65-9 S64-9 S67-9 S60-24
 S60-25 S11-1) ;
```

- - - - - - - - - - - - - - - - - - - -

```
(TEST1: ORDINARY VERSION WITH P BUILDING)

             X S O W B M P

 X1-1      X                 4....
 S0-1        S               7.......
 W3-1              W         1.
 S21-1       S               2..
 O0-1          O             2..
 S30-2       S               1.
 B3-1                B       1.
 M6-1                  M     4....
 S21-2       S               2..
 O0-2          O             2..
 S25-1       S               3...
 O2-1          O             1.
 S9-1        S               3...
 O0-3          O             4....
 S30-3       S               20....................
 O1-1          O             4....
 S9-2        S               4....
 O1-2          O             2..
 S9-3        S               5.....
 W4-1              W         1.
 M7-1                  M     4....
 S21-4       S               2..
```

```
O8-4        O       4....
S30-4      S        1.
B4-1          B     2..
M4-1           M    7.......
S21-5      S        2..
O8-5        O       4....
S30-5      S        1.
W4-2         W      1.
M7-2           M    4....
S210-1     S        3...
O8-6        O       4....
S30-6      S        1.
B4-3          B     2..
M9-1           M    5.....
S21-7      S        2..
O8-7        O       4....
S30-7      S        1.
W4-3         W      1.
M6-3           M    4....
S210-2     S        3...
O8-8        O       2..
S25-3      S        3...
O2-2        O       1.
S9-4       S        3...
O8-9        O       2..
S25-4      S        3...
O2-3        O       1.
S9-5       S        5.....
W3-2         W      1.
S21-10     S        2..
O8-10       O       2..
S30-8      S        1.
B4-5          B     1.
MSN-1          M    3...
S3-3       S        4....
B3-2          B     1.
MSN-2          M    1.
S3-4       S        25.........................
O1-3        O       2..
S9-6       S        4....
O1-4        O       4....
S21-11     S        2..
O8-11       O       2..
S25-6      S        3...
O2-4        O       1.
S9-7       S        5.....
B3-3          B     1.
M9-3           M    2..
S21-12     S        2..
O8-12       O       4....
PN-2-1          P   1.
S41-1      S        5.....
O1-5        O       4....
S9-8       S        28............................
O1-6        O       4....
S9-9       S        4....
O1-7        O       4....
S21-13     S        2..
O8-13       O       2..
S25-9      S        3...
O2-5        O       1.
S9-10      S        3...
O8-14       O       4....
S30-9      S        1.
W4-4         W      1.
M7-3           M    4....
S210-3     S        3...
O8-15       O       4....
S30-10     S        1.
B4-6          B     2..
M4-2           M    7.......
S210-1     S        3...
O8-16       O       4....
S35-1      S        23.......................
O1-8        O       4....
S9-11      S        4....
O1-9        O       4....
S21-17     S        2..
O8-17       O       4....
S30-11     S        1.
B4-8          B     2..
M1-2           M    7.......
S21-18     S        2..
```

```
O8-18       O       4....
S30-12     S        1.
W4-5         W      1.
M8-1           M    4....
S210-4     S        3...
O8-19       O       2..
S25-11     S        3...
O2-6        O       1.
S9-12      S        4....
O8-20       O       4....
S30-13     S        1.
B4-10         B     2..
M9-4           M    5.....
S21-21     S        2..
O8-21       O       2..
S25-12     S        3...
O2-7        O       1.
S9-14      S        3...
O8-22       O       2..
S25-13     S        3...
O2-8        O       1.
S9-15      S        5.....
B3-4          B     1.
M8-6           M    2..
S21-23     S        2..
O8-23       O       2..
S25-14     S        3...
O2-9        O       1.
S9-16      S        26..........................
O1-10       O       4....
S9-17      S        4....
O1-11       O       4....
S21-24     S        2..
O8-24       O       2..
S25-16     S        3...
O2-10       O       1.
S9-18      S        4....
O8-25       O       4....
S30-14     S        1.
W4-6         W      1.
M8-2           M    4....
S210-5     S        3...
O8-26       O       4....
S30-15     S        1.
B4-12         B     2..
M6-4           M    7.......
S21-27     S        2..
O8-27       O       2..
S25-17     S        3...
O2-11       O       1.
S9-19      S        3...
O8-28       O       3...
S25-18     S        3...
O2-12       O       1.
S9-20      S        3...
O8-29       O       4....
S30-16     S        4....
W3K-1        W      5.....
S21-30     S        2..
O8-30       O       4....
S36-1      S        25.........................
O1-12       O       4....
S9-21      S        4....
O1-13       O       4....
S21-31     S        2..
O8-31       O       2..
S25-20     S        3...
O2-13       O       1.
S9-22      S        5.....
B3-5          B     1.
M6-5           M    4....
S21-32     S        2..
O8-32       O       2..
S25-21     S        3...
O2-14       O       1.
S9-23      S        3...
O8-33       O       3...
S25-22     S        3...
O2-15       O       1.
S9-24      S        3...
O8-34       O       4....
PN-6-1          P   1.
S41-2      S        5.....
```

```
Q1-14      Q          4....
S9-25     S           28...........................
Q1-15      Q          4....
S9-26     S           4....
Q1-16      Q          4....
S9-27     S           5.....
B3-6           B      1.
M4-5             M    4....
S210-3    S           3...
Q8-35      Q          4....
PW-5-1             P  1.
S41-3     S           5.....
Q1-17      Q          4....
S21-35    S           2..
Q8-35      Q          2..
S25-26    S           3...
Q2-16      Q          1.
S9-28     S           28...........................
Q1-18      Q          4....
S9-29     S           4....
Q1-19      Q          4....
S21-37    S           2..
Q8-37      Q          2..
S25-28    S           3...
Q2-17      Q          1.
S9-30     S           5.....
B3-7           B      1.
M5-6             M    4....
S21-38    S           2..
Q8-38      Q          4....
PW-3-1             P  1.
S41-4     S           5.....
Q1-20      Q          4....
S21-39    S           2..
Q8-39      Q          2..
S25-30    S           3...
Q2-18      Q          1.
S9-31     S           3...
Q8-40      Q          4....
PW-8-1             P  1.
S41-5     S           5.....
Q1-21      Q          4....
S9-32     S           23.......................

PERCENTAGES OF FIRINGS OF EACH TYPE, OUT OF TOTAL 842
X 0
S 58........................................................
Q 26.........................
W 1.
B 2..
M 10..........
P 0;
```

Appendix E. <u>TRACES FOR THE OTHER TESTS</u>

TEST1: VERSION OF KPKEG WITH ALL DEPTHS DECREMENTING FROM MAX LEVEL

```
  ..  ..  ..  ..
..  BK  ..  ..
  ..  ..WP..  ..
..  ..  ..  ..
  ..  ..WK..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 5 W
1 MOVING WP FROM E6 TO E7 LEVEL 7
  LEVEL - 6 B
  LEVEL - 5 B
. 2 MOVING BK FROM C7 TO D8 LEVEL 6
  CAN'T MOVE C7 D8
. 3 MOVING BK FROM C7 TO D7 LEVEL 6
    LEVEL - 5 W
    LEVEL - FAIL DEPTH 3 W
    SUCCEED C7 D7 = S23
      ..  ..  ..  ..
    ..  ..BKWP  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..WK..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (E6 E7) (C7 D7)
  RETRACTING C7 D7
RETRACTING E6 E7
LEVEL - 4 W
4 MOVING WK FROM E4 TO E5 LEVEL 7
  LEVEL - 6 B
  LEVEL - 5 B
. 5 MOVING BK FROM C7 TO D8 LEVEL 6
    LEVEL - 5 W
. . 6 MOVING WP FROM E6 TO E7 LEVEL 7
      SUCCEED STRAT LEVEL 7 = B15
        ..  BK  ..  ..
      ..  ..  WP  ..
        ..  ..  ..  ..
      ..  ..  WK  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
      (E4 E5) (C7 D8) (E6 E7)
    RETRACTING E6 E7
    LEVEL - 4 W
. . 7 MOVING WK FROM E5 TO D6 LEVEL 7
      LEVEL - 6 B
      LEVEL - 5 B
. . . 8 MOVING BK FROM D8 TO E8 LEVEL 6
        LEVEL - 5 W
. . . . 9 MOVING WP FROM E6 TO E7 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
          LEVEL - FAIL DEPTH 6 B
          SUCCEED E6 E7 = S23
            ..  ..BK..  ..
          ..  ..  WP  ..
            ..  WK  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 D8) (E5 D6) (D8 E8) (E6 E7)
        RETRACTING E6 E7
      RETRACTING D8 E8
      LEVEL - FAIL DEPTH 4 B
      SUCCEED E5 D6 = S23
    RETRACTING E5 D6
  RETRACTING C7 D8
. 10 MOVING BK FROM C7 TO D7 LEVEL 8
  CAN'T MOVE C7 D7
. 11 MOVING BK FROM C7 TO C8 LEVEL 6
    LEVEL - 5 W
. . 12 MOVING WP FROM E6 TO E7 LEVEL 7
```

```
          LEVEL - 6 B
          LEVEL - 5 B
    . . . 13 MOVING BK FROM C8 TO D8 LEVEL 6
          CAN'T MOVE C8 D8
    . . . 14 MOVING BK FROM C8 TO D7 LEVEL 6
            LEVEL - 5 W
            LEVEL - FAIL DEPTH 5 W
            SUCCEED C8 D7 = S23
              ..  ..  ..  ..
            ..  ..BKWP  ..
              ..  ..  ..  ..
            ..  ..  WK  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
            (E4 E5) (C7 C8) (E6 E7) (C8 D7)
          RETRACTING C8 D7
        RETRACTING E6 E7
        LEVEL - 4 W
    . . 15 MOVING WK FROM E5 TO D6 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
    . . . 16 MOVING BK FROM C8 TO D8 LEVEL 6
            TERMINAL WIN FOR B = S36
              ..  BK  ..  ..
            ..  ..  ..  ..
              ..  WKWP..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
            (E4 E5) (C7 C8) (E5 D6) (C8 D8)
            LEVEL - 5 W
    . . . . 17 MOVING WP FROM E6 TO E7 LEVEL 7
              LEVEL - 6 B
              SUCCEED STRAT LEVEL 6 = B2A
                ..  BK  ..  ..
              ..  ..  WP  ..
                ..  WK  ..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
              (E4 E5) (C7 C8) (E5 D6) (C8 D8) (E6 E7)
            RETRACTING E6 E7
            LEVEL - 4 W
    . . . . 18 MOVING WK FROM D6 TO E7 LEVEL 7
            CAN'T MOVE D6 E7
    . . . . 19 MOVING WK FROM D6 TO D7 LEVEL 7
            CAN'T MOVE D6 D7
            LEVEL - FAIL DEPTH 5 W
            SUCCEED C8 D8 = S23
              ..  BK  ..  ..
            ..  ..  ..  ..
              ..  WKWP..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
            (E4 E5) (C7 C8) (E5 D6) (C8 D8)
          RETRACTING C8 D8
        RETRACTING E5 D6
    . . 20 MOVING WK FROM E5 TO F6 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
    . . . 21 MOVING BK FROM C8 TO D8 LEVEL 6
            LEVEL - 5 W
    . . . . 22 MOVING WP FROM E6 TO E7 LEVEL 7
              LEVEL - 6 B
              SUCCEED STRAT LEVEL 6 = B2A
                ..  BK  ..  ..
              ..  ..  WP  ..
                ..  ..  WK  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
              (E4 E5) (C7 C8) (E5 F6) (C8 D8) (E6 E7)
```

```
            RETRACTING E6 E7
            LEVEL - 4 W
    . . . . 23 MOVING WK FROM F6 TO E7 LEVEL 7
            CAN'T MOVE F6 E7
    . . . . 24 MOVING WK FROM F6 TO F7 LEVEL 7
              LEVEL - 6 B
              LEVEL - 5 B
    . . . . . 25 MOVING BK FROM D8 TO E8 LEVEL 6
              CAN'T MOVE D8 E8
              LEVEL - FAIL DEPTH 6 B
              SUCCEED F6 F7 = S23
                ..  BK  ..  ..
              ..  ..  ..WK..
                ..  ..WP..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
              (E4 E5) (C7 C8) (E5 F6) (C8 D8) (F6 F7)
            RETRACTING F6 F7
          RETRACTING C8 D8
    . . . 26 MOVING BK FROM C8 TO D7 LEVEL 6
          CAN'T MOVE C8 D7
          LEVEL - FAIL DEPTH 4 B
          SUCCEED E5 F6 = S23
            ..BK..  ..  ..
          ..  ..  ..  ..
            ..  ..WPWK  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 C8) (E5 F6)
        RETRACTING E5 F6
      RETRACTING C7 C8
      LEVEL - 4 B
    . 27 MOVING BK FROM C7 TO D8 LEVEL 6
        LEVEL - 5 W
    . . 28 MOVING WP FROM E6 TO E7 LEVEL 7
          SUCCEED STRAT LEVEL 7 = B15
            ..  BK  ..  ..
          ..  ..  WP  ..
            ..  ..  ..  ..
          ..  ..  WK  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 D8) (E6 E7)
        RETRACTING E6 E7
        LEVEL - 4 W
    . . 29 MOVING WK FROM E5 TO D6 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
    . . . 30 MOVING BK FROM D8 TO E8 LEVEL 6
            LEVEL - 5 W
    . . . . 31 MOVING WP FROM E6 TO E7 LEVEL 7
              LEVEL - 6 B
              LEVEL - 5 B
              LEVEL - FAIL DEPTH 6 B
              SUCCEED E6 E7 = S23
                ..  ..BK..  ..
              ..  ..  WP  ..
                ..  WK  ..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
                ..  ..  ..  ..
              ..  ..  ..  ..
              (E4 E5) (C7 D8) (E5 D6) (D8 E8) (E6 E7)
            RETRACTING E6 E7
          RETRACTING D8 E8
          LEVEL - 4 B
    . . . 32 MOVING BK FROM D8 TO E8 LEVEL 6
            LEVEL - 5 W
    . . . . 33 MOVING WP FROM E6 TO E7 LEVEL 7
              LEVEL - 6 B
              LEVEL - 5 B
              LEVEL - 4 B
              LEVEL - FAIL DEPTH 6 B
              SUCCEED E6 E7 = S23
          ..  ..BK..  ..
        ..  ..  WP  ..
          ..  WK  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 D8) (E5 D6) (D8 E8) (E6 E7)
      RETRACTING E6 E7
    RETRACTING D8 E8
. . . 34 MOVING BK FROM D8 TO E7 LEVEL 6
    CAN'T MOVE D8 E7
    LEVEL - FAIL DEPTH 4 B
    SUCCEED E5 D6 = S23
      ..  BK  ..  ..
    ..  ..  ..  ..
      ..  WKWP..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (E4 E5) (C7 D8) (E5 D6)
  RETRACTING E5 D6
RETRACTING C7 D8
. 35 MOVING BK FROM C7 TO D7 LEVEL 6
  CAN'T MOVE C7 D7
. 36 MOVING BK FROM C7 TO C8 LEVEL 6
    LEVEL - 5 W
. . 37 MOVING WP FROM E6 TO E7 LEVEL 7
      LEVEL - 6 B
      LEVEL - 5 B
. . . 38 MOVING BK FROM C8 TO D8 LEVEL 6
      CAN'T MOVE C8 D8
. . . 39 MOVING BK FROM C8 TO D7 LEVEL 6
        LEVEL - 5 W
        LEVEL - FAIL DEPTH 5 W
        SUCCEED C8 D7 = S23
          ..  ..  ..  ..
        ..  ..BKWP  ..
          ..  ..  ..  ..
        ..  ..  WK  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 C8) (E6 E7) (C8 D7)
      RETRACTING C8 D7
    RETRACTING E6 E7
    LEVEL - 4 W
. . 40 MOVING WK FROM E5 TO D6 LEVEL 7
      LEVEL - 6 B
      LEVEL - 5 B
. . . 41 MOVING BK FROM C8 TO D8 LEVEL 6
        TERMINAL WIN FOR B = S35
          ..  BK  ..  ..
        ..  ..  ..  ..
          ..  WKWP..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 C8) (E5 D6) (C8 D8)
        LEVEL - 5 W
. . . . 42 MOVING WP FROM E6 TO E7 LEVEL 7
          LEVEL - 6 B
          SUCCEED STRAT LEVEL 6 = B2A
            ..  BK  ..  ..
          ..  ..  WP  ..
            ..  WK  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 C8) (E5 D6) (C8 D8) (E6 E7)
        RETRACTING E6 E7
        LEVEL - 4 W
. . . . 43 MOVING WK FROM D6 TO E7 LEVEL 7
        CAN'T MOVE D6 E7
. . . . 44 MOVING WK FROM D6 TO D7 LEVEL 7
```

```
        CAN'T MOVE D6 D7
        LEVEL - FAIL DEPTH 5 W
        SUCCEED C8 D8 = S23
          ..  BK  ..  ..
        ..  ..  ..  ..
          ..  WKWP..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 C8) (E5 D6) (C8 D8)
      RETRACTING C8 D8
    RETRACTING E5 D6
. . 45 MOVING WK FROM E5 TO F6 LEVEL 7
      LEVEL - 6 B
      LEVEL - 5 B
. . . 46 MOVING BK FROM C8 TO D8 LEVEL 6
        LEVEL - 5 W
. . . . 47 MOVING WP FROM E6 TO E7 LEVEL 7
          LEVEL - 6 B
          SUCCEED STRAT LEVEL 6 = B2A
            ..  BK  ..  ..
          ..  ..  WP  ..
            ..  ..  WK  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 C8) (E5 F6) (C8 D8) (E6 E7)
        RETRACTING E6 E7
        LEVEL - 4 W
. . . . 48 MOVING WK FROM F6 TO E7 LEVEL 7
        CAN'T MOVE F6 E7
. . . . 49 MOVING WK FROM F6 TO F7 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
. . . . ... 50 MOVING BK FROM D8 TO E8 LEVEL 6
          CAN'T MOVE D8 E8
          LEVEL - FAIL DEPTH 6 B
          SUCCEED F6 F7 = S23
            ..  BK  ..  ..
          ..  ..  ..WK..
            ..  ..WP..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 C8) (E5 F6) (C8 D8) (F6 F7)
        RETRACTING F6 F7
      RETRACTING C8 D8
. . . 51 MOVING BK FROM C8 TO D7 LEVEL 6
      CAN'T MOVE C8 D7
      LEVEL - 4 B
. . . 52 MOVING BK FROM C8 TO D8 LEVEL 6
        LEVEL - 5 W
. . . . 53 MOVING WP FROM E6 TO E7 LEVEL 7
          LEVEL - 6 B
          SUCCEED STRAT LEVEL 6 = B2A
            ..  BK  ..  ..
          ..  ..  WP  ..
            ..  ..  WK  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 C8) (E5 F6) (C8 D8) (E6 E7)
        RETRACTING E6 E7
        LEVEL - 4 W
. . . . 54 MOVING WK FROM F6 TO E7 LEVEL 7
        CAN'T MOVE F6 E7
. . . . 55 MOVING WK FROM F6 TO F7 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
. . . . . 56 MOVING BK FROM D8 TO E8 LEVEL 6
          CAN'T MOVE D8 E8
          LEVEL - 4 B
. . . . . 57 MOVING BK FROM D8 TO E8 LEVEL 6
          CAN'T MOVE D8 E8
. . . . . 58 MOVING BK FROM D8 TO E7 LEVEL 6
```

```
          CAN'T MOVE D8 E7
          LEVEL - FAIL DEPTH 8 B
          SUCCEED F6 F7 = S23
            .. BK .. ..
          .. .. ..WK..
            .. ..WP.. ..
          .. .. .. ..
            .. .. .. ..
          .. .. .. ..
            .. .. .. ..
          .. .. .. ..
          (E4 E5) (C7 C8) (E5 F6) (C8 D8) (F6 F7)
        RETRACTING F6 F7
      RETRACTING C8 D8
. . . . 59 MOVING BK FROM C8 TO D7 LEVEL 6
      CAN'T MOVE C8 D7
. . . . 60 MOVING BK FROM C8 TO C7 LEVEL 6
        LEVEL - 5 W
. . . . . 61 MOVING WP FROM E6 TO E7 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
. . . . . . 62 MOVING BK FROM C7 TO D8 LEVEL 6
          CAN'T MOVE C7 D8
. . . . . . 63 MOVING BK FROM C7 TO D7 LEVEL 6
            LEVEL - 5 W
            LEVEL - FAIL DEPTH 7 W
            SUCCEED C7 D7 = S23
              .. .. .. ..
            .. ..BKWP ..
              .. .. WK ..
            .. .. .. ..
              .. .. .. ..
            .. .. .. ..
              .. .. .. ..
            .. .. .. ..
            (E4 E5) (C7 C8) (E5 F6) (C8 C7) (E6 E7) (C7 D7)
          RETRACTING C7 D7
        RETRACTING E6 E7
        LEVEL - 4 W
. . . . . 64 MOVING WK FROM F6 TO E7 LEVEL 7
          LEVEL - 6 B
          LEVEL - 5 B
. . . . . . 65 MOVING BK FROM C7 TO D8 LEVEL 6
          CAN'T MOVE C7 D8
. . . . . . 66 MOVING BK FROM C7 TO D7 LEVEL 6
          CAN'T MOVE C7 D7
. . . . . . 67 MOVING BK FROM C7 TO C8 LEVEL 6
            LEVEL - 5 W
. . . . . . . 68 MOVING WK FROM E7 TO E8 LEVEL 7
              TERMINAL WIN FOR W = S36
                ..BK..WK.. ..
              .. .. .. ..
                .. ..WP.. ..
              .. .. .. ..
                .. .. .. ..
              .. .. .. ..
                .. .. .. ..
              .. .. .. ..
              (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8) (E7 E8)
              LEVEL - 6 B
              LEVEL - 5 B
. . . . . . . . 69 MOVING BK FROM C8 TO D8 LEVEL 6
              CAN'T MOVE C8 D8
. . . . . . . . 70 MOVING BK FROM C8 TO D7 LEVEL 6
              CAN'T MOVE C8 D7
              LEVEL - FAIL DEPTH 8 B
              SUCCEED E7 E8 = S23
                ..BK..WK.. ..
              .. .. .. ..
                .. ..WP.. ..
              .. .. .. ..
                .. .. .. ..
              .. .. .. ..
                .. .. .. ..
              .. .. .. ..
              (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8) (E7 E8)
            RETRACTING E7 E8
          RETRACTING C7 C8
          LEVEL - 4 B
. . . . . 71 MOVING BK FROM C7 TO D8 LEVEL 6
          CAN'T MOVE C7 D8
. . . . . 72 MOVING BK FROM C7 TO D7 LEVEL 6
          CAN'T MOVE C7 D7
```

```
. . . . . 73 MOVING BK FROM C7 TO C8 LEVEL 6
            LEVEL - 5 W
. . . . . . 74 MOVING WK FROM E7 TO E8 LEVEL 7
              TERMINAL WIN FOR W = S36
                ..BK..WK.. ..
              .. .. .. ..
                .. ..WP.. ..
              .. .. .. ..
                .. .. .. ..
              .. .. .. ..
                .. .. .. ..
              .. .. .. ..
              (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8) (E7 E8)
              LEVEL - 6 B
              LEVEL - 5 B
. . . . . . . . 75 MOVING BK FROM C8 TO D8 LEVEL 6
              CAN'T MOVE C8 D8
. . . . . . . . 76 MOVING BK FROM C8 TO D7 LEVEL 6
              CAN'T MOVE C8 D7
              LEVEL - 4 B
. . . . . . . . 77 MOVING BK FROM C8 TO D8 LEVEL 6
              CAN'T MOVE C8 D8
. . . . . . . . 78 MOVING BK FROM C8 TO D7 LEVEL 6
              CAN'T MOVE C8 D7
. . . . . . . . 79 MOVING BK FROM C8 TO C7 LEVEL 6
                TERMINAL WIN FOR W = S36
                  .. ..WK.. ..
                .. BK .. ..
                  .. ..WP.. ..
                .. .. .. ..
                  .. .. .. ..
                .. .. .. ..
                  .. .. .. ..
                .. .. .. ..
                (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8) (E7 E8) (C8 C7)
              RETRACTING C8 C7
              LEVEL - FAIL DEPTH 8 B
              SUCCEED E7 E8 = S23
            RETRACTING E7 E8
          RETRACTING C7 C8
. . . . . 80 MOVING BK FROM C7 TO D6 LEVEL 6
          CAN'T MOVE C7 D6
          LEVEL - FAIL DEPTH 6 B
          SUCCEED F6 E7 = S23
            .. .. .. ..
          .. BK WK ..
            .. ..WP.. ..
          .. .. .. ..
            .. .. .. ..
          .. .. .. ..
            .. .. .. ..
          .. .. .. ..
          (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7)
        RETRACTING F6 E7
      RETRACTING C8 C7
      LEVEL - FAIL DEPTH 4 B
      SUCCEED E5 F6 = S23
    RETRACTING E5 F6
  RETRACTING C7 C8
. 81 MOVING BK FROM C7 TO D6 LEVEL 6
  CAN'T MOVE C7 D6
  LEVEL - FAIL DEPTH 2 B
  SUCCEED E4 E5 = S23
    .. .. .. ..
  .. BK .. ..
    .. ..WP.. ..
  .. .. WK ..
    .. .. .. ..
  .. .. .. ..
    .. .. .. ..
  .. .. .. ..
  (E4 E5)
MOVING (W WK E4 E5)
```

- - - - - - - - - - - - - - - - - - - -

TEST1: ORDINARY VERSION (DECR/INCR) WITH NO P BUILDING

```
  .. .. .. ..
.. BK .. ..
  .. ..WP.. ..
..  ..  ..  ..
  ..  ..WK..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 5 W
1 MOVING WP FROM E6 TO E7 LEVEL 5
. 2 MOVING BK FROM C7 TO D8 LEVEL 5
  CAN'T MOVE C7 D8
. 3 MOVING BK FROM C7 TO D7 LEVEL 5
    LEVEL + 6 W
    LEVEL + FAIL DEPTH 3 W
    SUCCEED C7 D7 = S23
      ..  ..  ..  ..
    ..  ..BKWP  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..WK..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (E6 E7) (C7 D7)
  RETRACTING C7 D7
RETRACTING E6 E7
LEVEL - 4 W
4 MOVING WK FROM E4 TO E5 LEVEL 4
. 5 MOVING BK FROM C7 TO D8 LEVEL 4
. . 6 MOVING WK FROM E5 TO D6 LEVEL 4
. . . 7 MOVING BK FROM D8 TO E8 LEVEL 4
. . . . 8 MOVING WK FROM D6 TO E7 LEVEL 4
        CAN'T MOVE D6 E7
. . . . 9 MOVING WK FROM D6 TO D7 LEVEL 4
        CAN'T MOVE D6 D7
        LEVEL + 5 W
. . . . 10 MOVING WP FROM E6 TO E7 LEVEL 4
          LEVEL + 5 B
          LEVEL + 6 B
          LEVEL + 7 B
          LEVEL + FAIL DEPTH 6 B
          SUCCEED E6 E7 = S23
            ..  ..BK..  ..
          ..  ..  WP  ..
            ..  WK  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 D8) (E5 D6) (D8 E8) (E6 E7)
        RETRACTING E6 E7
      RETRACTING D8 E8
. . . 11 MOVING BK FROM D8 TO E7 LEVEL 4
      CAN'T MOVE D8 E7
      LEVEL + 5 B
. . . 12 MOVING BK FROM D8 TO E8 LEVEL 4
. . . . 13 MOVING WK FROM D6 TO E7 LEVEL 5
        CAN'T MOVE D6 E7
. . . . 14 MOVING WK FROM D6 TO D7 LEVEL 5
        CAN'T MOVE D6 D7
        LEVEL + 5 W
. . . . 15 MOVING WP FROM E6 TO E7 LEVEL 5
          LEVEL + 6 B
          LEVEL + 7 B
          LEVEL + FAIL DEPTH 6 B
          SUCCEED E6 E7 = S23
            ..  ..BK..  ..
          ..  ..  WP  ..
            ..  WK  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 D8) (E5 D6) (D8 E8) (E6 E7)
        RETRACTING E6 E7
      RETRACTING D8 E8
      LEVEL + 6 B
      LEVEL + 7 B
      LEVEL + FAIL DEPTH 4 B
      SUCCEED E5 D6 = S23
    RETRACTING E5 D6
  RETRACTING C7 D8
. 16 MOVING BK FROM C7 TO D7 LEVEL 4
  CAN'T MOVE C7 D7
. 17 MOVING BK FROM C7 TO C8 LEVEL 4
. . 18 MOVING WK FROM E5 TO D6 LEVEL 4
. . . 19 MOVING BK FROM C8 TO D8 LEVEL 4
      TERMINAL WIN FOR B = S36
        ..  BK  ..  ..
      ..  ..  ..  ..
        ..  WKWP..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
      (E4 E5) (C7 C8) (E5 D6) (C8 D8)
      LEVEL + FAIL DEPTH 5 W
      SUCCEED C8 D8 = S23
    RETRACTING C8 D8
  RETRACTING E5 D6
. . 20 MOVING WK FROM E5 TO F6 LEVEL 4
. . . 21 MOVING BK FROM C8 TO D8 LEVEL 4
. . . . 22 MOVING WK FROM F6 TO E7 LEVEL 4
      CAN'T MOVE F6 E7
. . . . 23 MOVING WK FROM F6 TO F7 LEVEL 4
. . . . . 24 MOVING BK FROM D8 TO E8 LEVEL 4
        CAN'T MOVE D8 E8
. . . . . 25 MOVING BK FROM D8 TO E7 LEVEL 4
        CAN'T MOVE D8 E7
        LEVEL + 5 B
. . . . . 26 MOVING BK FROM D8 TO E8 LEVEL 4
        CAN'T MOVE D8 E8
        LEVEL + 6 B
        LEVEL + 7 B
        LEVEL + FAIL DEPTH 6 B
        SUCCEED F6 F7 = S23
          ..  BK  ..  ..
        ..  ..  ..WK..
          ..  ..WP..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 C8) (E5 F6) (C8 D8) (F6 F7)
      RETRACTING F6 F7
    RETRACTING C8 D8
. . . 27 MOVING BK FROM C8 TO D7 LEVEL 4
    CAN'T MOVE C8 D7
. . . 28 MOVING BK FROM C8 TO C7 LEVEL 4
. . . . 29 MOVING WK FROM F6 TO E7 LEVEL 4
. . . . . 30 MOVING BK FROM C7 TO D8 LEVEL 4
        CAN'T MOVE C7 D8
. . . . . 31 MOVING BK FROM C7 TO D7 LEVEL 4
        CAN'T MOVE C7 D7
. . . . . 32 MOVING BK FROM C7 TO C8 LEVEL 4
          LEVEL + 5 W
. . . . . . 33 MOVING WK FROM E7 TO E8 LEVEL 4
            TERMINAL WIN FOR W = S36
              ..BK..WK..  ..
            ..  ..  ..  ..
              ..  ..WP..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
            (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8) (E7 E8)
            LEVEL + FAIL DEPTH 8 B
            SUCCEED E7 E8 = S23
          RETRACTING E7 E8
        RETRACTING C7 C8
. . . . . 34 MOVING BK FROM C7 TO D6 LEVEL 4
        CAN'T MOVE C7 D6
        LEVEL + 5 B
. . . . . 35 MOVING BK FROM C7 TO D8 LEVEL 4
        CAN'T MOVE C7 D8
. . . . . 36 MOVING BK FROM C7 TO D7 LEVEL 4
        CAN'T MOVE C7 D7
. . . . . 37 MOVING BK FROM C7 TO C8 LEVEL 4
          LEVEL + 5 W
. . . . . . 38 MOVING WK FROM E7 TO E8 LEVEL 5
            TERMINAL WIN FOR W = S36
              ..BK..WK..  ..
            ..  ..  ..  ..
```

```
              ..  ..WP..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
              ..  ..  ..  ..
            ..  ..  ..  ..
            (E4 E5) (C7 C8) (E5 F6) (C8 C7) (F6 E7) (C7 C8) (E7 E8)
            LEVEL + FAIL DEPTH 8 B
            SUCCEED E7 E8 = S23
          RETRACTING E7 E8
        RETRACTING C7 C8
        LEVEL + 6 B
        LEVEL + 7 B
        LEVEL + FAIL DEPTH 6 B
        SUCCEED F6 E7 = S23
      RETRACTING F6 E7
    RETRACTING C8 C7
    LEVEL + 5 B
. . . 39 MOVING BK FROM C8 TO D8 LEVEL 4
. . . . 40 MOVING WK FROM F6 TO E7 LEVEL 5
      CAN'T MOVE F6 E7
. . . . 41 MOVING WK FROM F6 TO F7 LEVEL 5
. . . . . 42 MOVING BK FROM D8 TO E8 LEVEL 4
        CAN'T MOVE D8 E8
        LEVEL + 6 B
        LEVEL + 7 B
        LEVEL + FAIL DEPTH 6 B
        SUCCEED F6 F7 = S23
          ..  BK  ..  ..
        ..  ..  ..WK..
          ..  ..WP..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 C8) (E5 F6) (C8 D8) (F6 F7)
      RETRACTING F6 F7
    RETRACTING C8 D8
. . . 43 MOVING BK FROM C8 TO D7 LEVEL 4
    CAN'T MOVE C8 D7
    LEVEL + 6 B
    LEVEL + 7 B
    LEVEL + FAIL DEPTH 4 B
    SUCCEED E5 F6 = S23
      ..BK..  ..  ..
    ..  ..  ..  ..
      ..  ..WPWK  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (E4 E5) (C7 C8) (E5 F6)
  RETRACTING E5 F6
RETRACTING C7 C8
. 44 MOVING BK FROM C7 TO D6 LEVEL 4
CAN'T MOVE C7 D6
LEVEL + 5 B
. 45 MOVING BK FROM C7 TO D8 LEVEL 4
. . 46 MOVING WK FROM E5 TO D6 LEVEL 5
. . . 47 MOVING BK FROM D8 TO E8 LEVEL 4
. . . . 48 MOVING WK FROM D6 TO E7 LEVEL 5
      CAN'T MOVE D6 E7
. . . . 49 MOVING WK FROM D6 TO D7 LEVEL 5
      CAN'T MOVE D6 D7
      LEVEL + 5 W
. . . . 50 MOVING WP FROM E6 TO E7 LEVEL 5
        LEVEL + 6 B
        LEVEL + 7 B
        LEVEL + FAIL DEPTH 6 B
        SUCCEED E6 E7 = S23
          ..  ..BK..  ..
        ..  ..  WP  ..
          ..  WK  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 D8) (E5 D6) (D8 E8) (E6 E7)
      RETRACTING E6 E7
    RETRACTING D8 E8
```

```
      LEVEL + 6 B
      LEVEL + 7 B
      LEVEL + FAIL DEPTH 4 B
      SUCCEED E5 D6 = S23
    RETRACTING E5 D6
  RETRACTING C7 D8
. 51 MOVING BK FROM C7 TO D7 LEVEL 4
  CAN'T MOVE C7 D7
. 52 MOVING BK FROM C7 TO C8 LEVEL 4
. . 53 MOVING WK FROM E5 TO D6 LEVEL 5
. . . 54 MOVING BK FROM C8 TO D8 LEVEL 4
        TERMINAL WIN FOR B = S35
          ..  BK  ..  ..
        ..  ..  ..  ..
          ..  WKWP..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
          ..  ..  ..  ..
        ..  ..  ..  ..
        (E4 E5) (C7 C8) (E5 D6) (C8 D8)
        LEVEL + FAIL DEPTH 5 W
        SUCCEED C8 D8 = S23
      RETRACTING C8 D8
    RETRACTING E5 D6
. . 55 MOVING WK FROM E5 TO F6 LEVEL 5
. . . 56 MOVING BK FROM C8 TO D8 LEVEL 4
. . . . 57 MOVING WK FROM F6 TO E7 LEVEL 5
        CAN'T MOVE F6 E7
. . . . 58 MOVING WK FROM F6 TO F7 LEVEL 5
. . . . . 59 MOVING BK FROM D8 TO E8 LEVEL 4
          CAN'T MOVE D8 E8
          LEVEL + 6 B
          LEVEL + 7 B
          LEVEL + FAIL DEPTH 6 B
          SUCCEED F6 F7 = S23
            ..  BK  ..  ..
          ..  ..  ..WK..
            ..  ..WP..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
            ..  ..  ..  ..
          ..  ..  ..  ..
          (E4 E5) (C7 C8) (E5 F6) (C8 D8) (F6 F7)
        RETRACTING F6 F7
      RETRACTING C8 D8
. . . 60 MOVING BK FROM C8 TO D7 LEVEL 4
      CAN'T MOVE C8 D7
      LEVEL + 6 B
      LEVEL + 7 B
      LEVEL + FAIL DEPTH 4 B
      SUCCEED E5 F6 = S23
        ..BK..  ..  ..
      ..  ..  ..  ..
        ..  ..WPWK  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
      (E4 E5) (C7 C8) (E5 F6)
    RETRACTING E5 F6
  RETRACTING C7 C8
  LEVEL + 6 B
  LEVEL + 7 B
  LEVEL + FAIL DEPTH 2 B
  SUCCEED E4 E5 = S23
MOVING (W WK E4 E5)
```

- - - - - - - - - - - - - - - - - - - -

TEST2 FINAL RUN

```
  ..  ..BK..  ..
..  ..  ..  ..
  ..  ..WK..  ..
..  ..  WP  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
```

E.

```
.. .. .. ..
LEVEL - 5 W
1 MOVING WK FROM E6 TO E7 LEVEL 5
CAN'T MOVE E6 E7
2 MOVING WK FROM E6 TO D7 LEVEL 5
CAN'T MOVE E6 D7
3 MOVING WK FROM E6 TO F7 LEVEL 5
CAN'T MOVE E6 F7
4 MOVING WK FROM E6 TO D6 LEVEL 5
  TERMINAL WIN FOR W = S36R
    ..  ..BK..  ..
  ..  ..  ..  ..
    ..  WK  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E6 D6)
  LEVEL + FAIL DEPTH 2 B
  SUCCEED E6 D6 = S23
  ADDPROD PN-1 DEPTH 1 LEVEL 5 E6 D6
MOVING (W WK E6 D6)
  ..  ..BK..  ..
..  ..  ..  ..
  ..  WK  ..  ..
..  ..  WP  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 6 B
LEVEL - 5 B
LEVEL - 4 B
1 MOVING BK FROM E8 TO E7 LEVEL 4
CAN'T MOVE E8 E7
2 MOVING BK FROM E8 TO D7 LEVEL 4
CAN'T MOVE E8 D7
3 MOVING BK FROM E8 TO F7 LEVEL 4
  TERMINAL WIN FOR W = S36R
    ..  ..  ..  ..
  ..  ..  ..BK..
    ..  WK  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E8 F7)
RETRACTING E8 F7
LEVEL - 3 B
4 MOVING BK FROM E8 TO E7 LEVEL 3
CAN'T MOVE E8 E7
5 MOVING BK FROM E8 TO D7 LEVEL 3
CAN'T MOVE E8 D7
6 MOVING BK FROM E8 TO F7 LEVEL 3
  TERMINAL WIN FOR W = S36R
    ..  ..  ..  ..
  ..  ..  ..BK..
    ..  WK  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E8 F7)
RETRACTING E8 F7
LEVEL - 2 B
7 MOVING BK FROM E8 TO D8 LEVEL 2
  TERMINAL WIN FOR W = S36R
    ..  BK  ..  ..
  ..  ..  ..  ..
    ..  WK  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E8 D8)
RETRACTING E8 D8
8 MOVING BK FROM E8 TO D8 LEVEL 2
  TERMINAL WIN FOR W = S36R
    ..  BK  ..  ..
```

```
  ..  ..  ..  ..
    ..  WK  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E8 D8)
RETRACTING E8 D8
9 MOVING BK FROM E8 TO E7 LEVEL 2
CAN'T MOVE E8 E7
10 MOVING BK FROM E8 TO D7 LEVEL 2
CAN'T MOVE E8 D7
LEVEL - 1 B
11 MOVING BK FROM E8 TO F8 LEVEL 1
  TERMINAL WIN FOR W = S36R
    ..  ..  BK  ..
  ..  ..  ..  ..
    ..  WK  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E8 F8)
RETRACTING E8 F8
LEVEL - FAIL DEPTH 1 B
12 MOVING BK FROM E8 TO D8 LEVEL 0
  ..  BK  ..  ..
..  ..  ..  ..
  ..  WK  ..  ..
..  ..  WP  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 5 W
1 MOVING WP FROM E5 TO E6 LEVEL 5
. 2 MOVING BK FROM D8 TO E8 LEVEL 6
. . 3 MOVING WP FROM E6 TO E7 LEVEL 5
      LEVEL + 6 B
      LEVEL + 7 B
      LEVEL + FAIL DEPTH 4 B
      SUCCEED E6 E7 = S23
        ..  ..BK..  ..
      ..  ..  WP  ..
        ..  WK  ..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
      (E5 E6) (D8 E8) (E6 E7)
      ADDPROD PN-1 DEPTH 3 LEVEL 5 E6 E7
    RETRACTING E6 E7
  RETRACTING D8 E8
  LEVEL + 6 B
  LEVEL + 7 B
  LEVEL + FAIL DEPTH 2 B
  SUCCEED E5 E6 = S23
MOVING (W WP E5 E6)
  ..  BK  ..  ..
..  ..  ..  ..
  ..  WKWP..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 6 B
LEVEL - 5 B
1 MOVING BK FROM D8 TO E8 LEVEL 5
. 2 MOVING WP FROM E6 TO E7 LEVEL 6
    LEVEL + 6 B
    LEVEL + 7 B
    LEVEL + FAIL DEPTH 3 B
    SUCCEED E6 E7 = S23
      ..  ..BK..  ..
    ..  ..  WP  ..
      ..  WK  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
```

E.

```
      ..  ..  ..  ..
    ..  ..  ..  ..
    (D8 E8) (E6 E7)
  RETRACTING E6 E7
RETRACTING D8 E8
LEVEL - 4 B
3 MOVING BK FROM D8 TO E8 LEVEL 4
. 4 MOVING WK FROM D6 TO E7 LEVEL 4
  CAN'T MOVE D6 E7
. 5 MOVING WK FROM D6 TO D7 LEVEL 4
  CAN'T MOVE D6 D7
  LEVEL + 5 W
. 6 MOVING WP FROM E6 TO E7 LEVEL 4
    LEVEL + 5 B
    LEVEL + 6 B
    LEVEL + 7 B
    LEVEL + FAIL DEPTH 3 B
    SUCCEED E6 E7 = S23
      ..  ..BK..  ..
    ..  ..  WP  ..
      ..  WK  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (D8 E8) (E6 E7)
  RETRACTING E6 E7
RETRACTING D8 E8
7 MOVING BK FROM D8 TO E7 LEVEL 4
CAN'T MOVE D8 E7
LEVEL - 3 B
8 MOVING BK FROM D8 TO E8 LEVEL 3
  LEVEL + 4 W
. 9 MOVING WK FROM D6 TO E7 LEVEL 3
  CAN'T MOVE D6 E7
. 10 MOVING WK FROM D6 TO D7 LEVEL 3
  CAN'T MOVE D6 D7
  LEVEL + 5-W
. 11 MOVING WP FROM E6 TO E7 LEVEL 3
. . 12 MOVING BK FROM E8 TO E7 CAPTURING WP LEVEL 5
    CAN'T MOVE E8 E7
    LEVEL + 4 B
    LEVEL + 5 B
    LEVEL + 6 B
    LEVEL + 7 B
    LEVEL + FAIL DEPTH 3 B
    SUCCEED E6 E7 = S23
      ..  ..BK..  ..
    ..  ..  WP  ..
      ..  WK  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (D8 E8) (E6 E7)
  RETRACTING E6 E7
RETRACTING D8 E8
13 MOVING BK FROM D8 TO E7 LEVEL 3
CAN'T MOVE D8 E7
14 MOVING BK FROM D8 TO D7 LEVEL 3
CAN'T MOVE D8 D7
LEVEL - 2 B
15 MOVING BK FROM D8 TO E7 LEVEL 2
CAN'T MOVE D8 E7
16 MOVING BK FROM D8 TO D7 LEVEL 2
CAN'T MOVE D8 D7
17 MOVING BK FROM D8 TO C7 LEVEL 2
CAN'T MOVE D8 C7
LEVEL - 1 B
18 MOVING BK FROM D8 TO C8 LEVEL 1
  LEVEL + 2 W
. 19 MOVING WK FROM D6 TO C6 LEVEL 1
. . 20 MOVING BK FROM C8 TO B8 LEVEL 2
. . . 21 MOVING WK FROM C6 TO C7 LEVEL 1
      CAN'T MOVE C6 C7
. . . 22 MOVING WK FROM C6 TO B7 LEVEL 1
      CAN'T MOVE C6 B7
. . . 23 MOVING WK FROM C6 TO B6 LEVEL 1
. . . . 24 MOVING BK FROM B8 TO A8 LEVEL 2
. . . . . 25 MOVING WK FROM B6 TO A6 LEVEL 1
            LEVEL + 2 B
```

```
. . . . . . . 26 MOVING BK FROM A8 TO B7 LEVEL 2
              CAN'T MOVE A8 B7
. . . . . . . 27 MOVING BK FROM A8 TO A7 LEVEL 2
              CAN'T MOVE A8 A7
              LEVEL + 3 B
. . . . . . . 28 MOVING BK FROM A8 TO B8 LEVEL 2
. . . . . . . . 29 MOVING WK FROM A6 TO B6 LEVEL 3
                  SUCCEED A6 B6 = S23
                    BK  ..  ..  ..
                  ..  ..  ..  ..
                    WK  ..WP..  ..
                  ..  ..  ..  ..
                    ..  ..  ..  ..
                  ..  ..  ..  ..
                    ..  ..  ..  ..
                  ..  ..  ..  ..
                  (D8 C8) (D6 C6) (C8 B8) (C6 B6) (B8 A8) (B6 A6) (A8 B8) (A6 B6)
                RETRACTING A6 B6
              RETRACTING A8 B8
. . . . . . . 30 MOVING BK FROM A8 TO B7 LEVEL 2
              CAN'T MOVE A8 B7
. . . . . . . 31 MOVING BK FROM A8 TO A7 LEVEL 2
              CAN'T MOVE A8 A7
              LEVEL + 4 B
. . . . . . . 32 MOVING BK FROM A8 TO B8 LEVEL 2
. . . . . . . . 33 MOVING WK FROM A6 TO B7 LEVEL 4
                CAN'T MOVE A6 B7
. . . . . . . . 34 MOVING WK FROM A6 TO B6 LEVEL 4
                  SUCCEED A6 B6 = S23
                    BK  ..  ..  ..
                  ..  ..  ..  ..
                    WK  ..WP..  ..
                  ..  ..  ..  ..
                    ..  ..  ..  ..
                  ..  ..  ..  ..
                    ..  ..  ..  ..
                  ..  ..  ..  ..
                  (D8 C8) (D6 C6) (C8 B8) (C6 B6) (B8 A8) (B6 A6) (A8 B8) (A6 B6)
         ..
          .
          .
```

- -

```
TEST3 WITH P BUILDING ON ONLY PART OF THE TIME DUE TO BUG

  ..  ..BK..  ..
..  ..  ..  ..
  ..  ..WK..  ..
..  ..  ..  ..
  ..  ..WP..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 5 W
1 MOVING WP FROM E4 TO E5 LEVEL 5
  TERMINAL WIN FOR W = 5360
    ..  ..BK..  ..
  ..  ..  ..  ..
    ..  ..WK..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E4 E5)
  LEVEL + FAIL DEPTH 2 B
  SUCCEED E4 E5 = S23
  ADDPROD PN-1 DEPTH 1 LEVEL 5 E4 E5
MOVING (W WP E4 E5)
  ..  ..BK..  ..
..  ..  ..  ..
  ..  ..WK..  ..
..  ..  WP  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 6 B
LEVEL - 5 B
LEVEL - 4 B
```

```
1 MOVING BK FROM E8 TO E7 LEVEL 4
CAN'T MOVE E8 E7
2 MOVING BK FROM E8 TO D7 LEVEL 4
CAN'T MOVE E8 D7
3 MOVING BK FROM E8 TO F7 LEVEL 4
CAN'T MOVE E8 F7
LEVEL - 3 B
4 MOVING BK FROM E8 TO E7 LEVEL 3
CAN'T MOVE E8 E7
5 MOVING BK FROM E8 TO D7 LEVEL 3
CAN'T MOVE E8 D7
6 MOVING BK FROM E8 TO F7 LEVEL 3
CAN'T MOVE E8 F7
LEVEL - 2 B
7 MOVING BK FROM E8 TO E7 LEVEL 2
CAN'T MOVE E8 E7
8 MOVING BK FROM E8 TO D7 LEVEL 2
CAN'T MOVE E8 D7
9 MOVING BK FROM E8 TO F7 LEVEL 2
CAN'T MOVE E8 F7
LEVEL - 1 B
10 MOVING BK FROM E8 TO D8 LEVEL 1
  TERMINAL WIN FOR W = S36R
    ..  BK  ..  ..
  ..  ..  ..  ..
    ..  ..WK..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E8 D8)
RETRACTING E8 D8
11 MOVING BK FROM E8 TO F8 LEVEL 1
  TERMINAL WIN FOR W = S36R
    ..  ..  BK  ..
  ..  ..  ..  ..
    ..  ..WK..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E8 F8)
RETRACTING E8 F8
LEVEL - FAIL DEPTH 1 B
1 MOVING BK FROM E8 TO F8 LEVEL 1
  ..  ..  BK  ..
..  ..  ..  ..
  ..  ..WK..  ..
..  ..  WP  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
LEVEL - 5 W
1 MOVING WK FROM E6 TO E7 LEVEL 5
CAN'T MOVE E6 E7
2 MOVING WK FROM E6 TO D7 LEVEL 5
. 3 MOVING BK FROM F8 TO E8 LEVEL 5
  CAN'T MOVE F8 E8
  LEVEL + 6 B
  LEVEL + 7 B
  LEVEL + FAIL DEPTH 2 B
  SUCCEED E6 D7 = S23
    ..  ..  BK  ..
  ..  ..WK..  ..
    ..  ..  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  (E6 D7)
MOVING (W WK E6 D7)
  ..  ..  BK  ..
..  ..WK..  ..
  ..  ..  ..  ..
..  ..  WP  ..
  ..  ..  ..  ..
..  ..  ..  ..
  ..  ..  ..  ..
..  ..  ..  ..
```

```
LEVEL - 6 B
LEVEL - 5 B
1 MOVING BK FROM F8 TO E8 LEVEL 5
CAN'T MOVE F8 E8
LEVEL - 4 B
2 MOVING BK FROM F8 TO E8 LEVEL 4
CAN'T MOVE F8 E8
3 MOVING BK FROM F8 TO E7 LEVEL 4
CAN'T MOVE F8 E7
4 MOVING BK FROM F8 TO F7 LEVEL 4
  LEVEL + 5 W
. 5 MOVING WP FROM E5 TO E6 LEVEL 4
. . 6 MOVING BK FROM F7 TO E8 LEVEL 5
    CAN'T MOVE F7 E8
. . 7 MOVING BK FROM F7 TO E7 LEVEL 5
    CAN'T MOVE F7 E7
    LEVEL + 5 B
. . 8 MOVING BK FROM F7 TO E8 LEVEL 5
    CAN'T MOVE F7 E8
    LEVEL + 6 B
    LEVEL + 7 B
    LEVEL + FAIL DEPTH 3 B
    SUCCEED E5 E6 = S23
      ..  ..  ..  ..
    ..  ..WK..BK..
      ..  ..WP..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (F8 F7) (E5 E6)
  RETRACTING E5 E6
RETRACTING F8 F7
LEVEL - 3 B
9 MOVING BK FROM F8 TO E8 LEVEL 3
CAN'T MOVE F8 E8
10 MOVING BK FROM F8 TO E7 LEVEL 3
CAN'T MOVE F8 E7
11 MOVING BK FROM F8 TO F7 LEVEL 3
. 12 MOVING WK FROM D7 TO E7 LEVEL 3
  CAN'T MOVE D7 E7
. 13 MOVING WK FROM D7 TO E6 LEVEL 3
  CAN'T MOVE D7 E6
. 14 MOVING WK FROM D7 TO D6 LEVEL 3
    TERMINAL WIN FOR W = S36R
      ..  ..  ..  ..
    ..  ..  ..BK..
      ..  WK  ..  ..
    ..  ..  WP  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (F8 F7) (D7 D6)
    LEVEL + FAIL DEPTH 3 B
    SUCCEED D7 D6 = S23
  RETRACTING D7 D6
RETRACTING F8 F7
LEVEL - 2 B
15 MOVING BK FROM F8 TO E8 LEVEL 2
CAN'T MOVE F8 E8
16 MOVING BK FROM F8 TO E7 LEVEL 2
CAN'T MOVE F8 E7
17 MOVING BK FROM F8 TO F7 LEVEL 2
  LEVEL + 3 W
. 18 MOVING WK FROM D7 TO E7 LEVEL 2
  CAN'T MOVE D7 E7
. 19 MOVING WK FROM D7 TO E6 LEVEL 2
  CAN'T MOVE D7 E6
. 20 MOVING WK FROM D7 TO D6 LEVEL 2
    TERMINAL WIN FOR W = S36R
      ..  ..  ..  ..
    ..  ..  ..BK..
      ..  WK  ..  ..
    ..  ..  WP  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (F8 F7) (D7 D6)
    LEVEL + FAIL DEPTH 3 B
    SUCCEED D7 D6 = S23
    RETRACTING D7 D6
  RETRACTING F8 F7
  LEVEL - 1 B
  21 MOVING BK FROM F8 TO G8 LEVEL 1
    LEVEL + 2 W
    LEVEL + 3 W
  . 22 MOVING WK FROM D7 TO E7 LEVEL 1
      TERMINAL WIN FOR W = S36
        ..  ..  ..BK..
      ..  ..  WK  ..
        ..  ..  ..  ..
      ..  ..  WP  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
      (F8 G8) (D7 E7)
      LEVEL + FAIL DEPTH 3 B
      SUCCEED D7 E7 = S23
    RETRACTING D7 E7
  RETRACTING F8 G8
  23 MOVING BK FROM F8 TO G7 LEVEL 1
    LEVEL + 2 W
    LEVEL + 3 W
  . 24 MOVING WK FROM D7 TO E7 LEVEL 1
      TERMINAL WIN FOR W = S36
        ..  ..  ..  ..
      ..  ..  WK  BK
        ..  ..  ..  ..
      ..  ..  WP  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
        ..  ..  ..  ..
      ..  ..  ..  ..
      (F8 G7) (D7 E7)
      LEVEL + FAIL DEPTH 3 B
      SUCCEED D7 E7 = S23
    RETRACTING D7 E7
  RETRACTING F8 G7
  LEVEL - FAIL DEPTH 1 B
  25 MOVING BK FROM F8 TO F7 LEVEL 1
    ..  ..  ..  ..
  ..  ..WK..BK..
    ..  ..  ..  ..
  ..  ..  WP  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
    ..  ..  ..  ..
  ..  ..  ..  ..
  LEVEL - 5 W
  1 MOVING WP FROM E5 TO E6 LEVEL 5
  . 2 MOVING BK FROM F7 TO E8 LEVEL 5
    CAN'T MOVE F7 E8
    LEVEL + 6 B
    LEVEL + 7 B
    LEVEL + FAIL DEPTH 2 B
    SUCCEED E5 E6 = S23
      ..  ..  ..  ..
    ..  ..WK..BK..
      ..  ..WP..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
      ..  ..  ..  ..
    ..  ..  ..  ..
    (E5 E6)
  MOVING (W WP E5 E6)

  RUN TIME 10 MIN. 59.5 SEC

  EXAM     TRY      FIRE     WMACT   E/F     E/T     T/F
  5836     2520     785      3407    7.43    2.31    3.22
  0.113    0.261    0.840    0.194   SEC AVG

  1787 INSERTS 1620 DELETES 445 WARNINGS 0 NEW OBJECTS
  MAX :SMPX LENGTH 118
  CORE (FREE.FULL): (6910 . 2470) USED (6160 . 562)

  FIRED 57 OUT OF 143 PRODS
```

Chapter VI

MiliPS/WBlox

A Natural Language Input Toy Blocks Problem Solver

Abstract. The MiliPS/WBlox production system is a combination of two major systems, one for processing a simple subset of natural language and the other for solving problems in a simple toy blocks domain. The emphasis of the natural language part is to study some problems of ambiguity and to illustrate a direct, non-syntactic-parsing approach to understanding natural language. The blocks problem solver deals with simple blocks manipulations, but deals with them in a general way. It features a simple goal-subgoal mechanism and conventions that allow choicepoints for a backtracking search. The blocks manipulations are a close imitation of Winograd's Planner system.

Table of Contents

For Chapter VI

A. Introduction

MiliPS is a production system (PS)● implementation of an extension of MILISY (<u>mini</u>-<u>linguistic</u> <u>sy</u>stem), a mini-program used to illustrate natural language processing in the CMU AI course. MILISY takes in facts about a toy blocks scene in restricted natural language, builds up a database of those facts, and answers queries about them. This chapter presents MiliPS in two versions. The first version, consisting of MiliPS alone, augments the language-processing aspects of MILISY, while the second, consisting of a further augmentation of MiliPS plus another system, WBlox (W for Winograd), emphasizes block-manipulation problem-solving aspects.

MiliPS aims to make the language processing more complete than MILISY, in being able to give information on and query more features of the blocks scene. The language that MiliPS understands is composed of descriptive attribute values (adjectives), nouns, main sentence function words, prepositional phrases representating relations between objects, and subordinate clauses that can be used to further refine descriptions of objects. This language can be expressed as an ambiguous context-free grammar, but MiliPS does not proceed by extracting the grammatical structure of its input as a parse tree. Ambiguities are resolved by flexible use of features of the scene, essentially as soon in the process of scanning the input as is logically possible.

The blocks manipulations that constitute WBlox are based closely on the problem-solving part of Winograd's SHRDLU program (Winograd, 1972). That subsystem of SHRDLU was coded in Micro-Planner (Sussman and Winograd, 1970; henceforth, referred to as Planner), a language specifically designed to make certain heuristic search operations automatic. WBlox moves single objects (rectangular blocks and pyramids) between locations in the scene without spatial rotation, finds locations to put them, builds stacks of them, and packs them compactly into a space if necessary. WBlox uses a hierarchical goal-subgoal structure to break big operations down into more primitive ones, with a set of indivisible primitives consisting of moving the hand to specific locations, grasping objects, and letting go of objects. At certain key points in the problem solution process, arbitrary choices are made, requiring WBlox to record its choice and the context, so that corrections are possible later in response to unforeseen difficulties. The particular approach to the search through the space of choices in WBlox is intended to imitate the Planner approach, not to represent the best scheme for PSs, which it certainly isn't.

The toy blocks domain has features that are abstractions of a much more general domain of discourse. It is composed of <u>objects</u> that have certain non-changeable <u>attributes</u>, and that enter into <u>relations</u> with other objects. This certainly models (abstractly) the physical world in which humans move, but it also goes much further, representing important aspects of human sociocultural organization, of economic systems, and of numerous more abstract formal (or informal) disciplines such as computer programming. (A piece of a computer program has attributes, e.g. what it is intended to do, and relations, e.g. dependence on other code for its inputs; there can be several pieces of code competing for the same space within a "block" of computer storage, etc.) Of

● PS will abbreviate production system, plural PSs; P will abbreviate production, plural Ps.

course, how relationships and attributes are structured in real domains does not correspond to how they are treated in toy blocks, but it is to be hoped that some of the more general techniques that work with a blocks domain might carry over, requiring only modification of the detailed semantics of specific relations and attributes.

That correspondence to more important problems provides some motivation for pursuing the present study. More motivation comes from the desire to develop a flexible PS-based approach to natural language processing, and to test its feasibility on a significant and classical AI task. WBlox also provides the opportunity to compare a PS program to a functionally similar one written in Planner. It may also provide future comparisons to other AI programming systems and proposals, and act as a benchmark.

For those familiar with Winograd's (1972) program, I will summarize the primary differences between the MiliPS/WBlox system and SHRDLU. The blocks part of SHRDLU has direct analogs in WBlox, except that WBlox doesn't do quite all of the bookkeeping and memory functions (such as remembering all the steps of a plan so they can be "executed" at the end of planning). This only means that MiliPS can't answer questions about why it did various steps in performing a particular command, and when it did them. The language understanding part of SHRDLU is much more capable than MiliPS. The internal representation is not as rich in MiliPS, especially in semantic attributes, e.g. "manipulable", and the language doesn't give full access to featues of the representation that it does have, like size and location. It recognizes only the imperative form of verbs, and can't deal with other more descriptive references to the commands that it can do. It doesn't interact to resolve ambiguities as SHRDLU did, but simply gives an error message and waits for a corrected version of the sentence. It is unable to dynamically define new words as SHRDLU was apparently able to do. Finally, there is very little in the way of language generation. Its replies are mostly fixed, and the ones that aren't fixed are descriptive, giving (stupidly) all the attributes' values for an object or all the relations it has with other objects, in order to tell the user about the object. On the other hand, it is quite capable of handling most of the ambiguities and reference problems that SHRDLU did, except references to objects in other sentences of a conversation, using, e.g., pronouns. It has captured many desirable features that go with a problem-solving system such as WBlox, and is a satisfactory first approximation.

The approach here has been in a way opposite to Winograd's. MiliPS started out as a comparison of PSs to MILISY, a program with very modest aspirations and serious deficiencies in dealing with its model of the blocks scene. MiliPS first overcame those deficiencies, and went rather far beyond any conceivable extensions of MILISY within its own control structure, which was a more traditional phrase-structure transformational one. Any comparison of PSs with that structure is not possible now because it would require either large extrapolations in MILISY's abilities or actually trying to extend the implementation to compare with MiliPS. After MiliPS had supposedly been refined to a stable version, the blocks manipulation task came along, and the urge to use MiliPS as an interface to a blocks problem solver was irresistable. But only a minimal sort of extension to MiliPS could be justified since the blocks manipulations were more central to the goals of investigating the properties of PSs. Thus the language is only a convenience in the final MiliPS/WBlox system. Winograd on the other hand concentrated on linguistic issues, and tacked on the blocks program as an easy means toward illustrating the power of his linguistic understanding system.

The structure of this chapter reflects the dual history and forced juxtaposition of two lines of research. Section B and Section C are devoted solely to describing MiliPS: its overall structure, the input language, how the language deals with describing all the desired features of the limited blocks scene, and the system it uses to disambiguate complex descriptions. The latter section gives more complete details of the actual PS structure. Section D and Section E do corresponding things for the WBlox system, touching only in passing the nature of the extensions to MiliPS that were required. Near the beginnings of both descriptions, some typical sentences and behaviors are discussed.

B. Overview of MiliPS

This section gives a general overview of MiliPS, postponing details until the next section. Section B.1 first discusses a few of the tests given to the program, with only vague descriptions of the processing done. It then gives a precise description of the task domain, including a grammar for the input language and a systematic presentation of semantic capabilities. Section B.2 uses very abstract Ps to describe the way the program works and outlines the processing of an input. Several levels of semantic processing are distinguished. Section B.3 discusses PS control and organization, low-level PS features, representation, and the expected extensibility of the present approach to syntax and semantics.

B.1. Features of the task

MiliPS has been tested on a set of 25 sentences, forming a continuous conversation about a single growing scene. The full dialog is given in Appendix C, along with trace information that will be explained in Section C. The following sentences will give the reader some idea of its capabilities.

MiliPS starts out with no initial scene, building up everything from descriptions of a scene by the user.

INPUT 1: (A LARGE GREEN BLOCK IS ON A TABLE)

In response to the first part of 1, MiliPS creates a block, adds "size large" and "color green" to its internal representation. It creates a table after scanning the rest of the input, and adds "color red" to its representation. Finally, it notes the relation "on" between the two new objects.

REPLY 1: (OKAY)

MiliPS indicates with the first reply that it has used everything in the input and hasn't noticed any unresolved ambiguities, inconsistencies, etc.

In three test sentences (not shown) MiliPS has been told about a ball on the block, and is able to determine that the description in 5 refers to that particular ball.

INPUT 5: (THE BALL ON THE BLOCK IS SMALL)

The relation "on the block" is necessary because there is a second ball in the scene. The effect here is to add "size small" to the internal representation for the ball.

REPLY 5: (OKAY)

The first five inputs describe a scene, and the next five primarily ask questions on that scene.

INPUT 7: (WHAT IS BLUE)

The query asks for all objects that have the color blue. MiliPS processes "what" by forming a set of all the objects in the scene; "what" is essentially a very ambigous noun phrase. Then it applies any further predicates in the sentence as restrictions to that set, and if anything is left when the end of the sentence is reached, it describes it as its answer.

REPLY 7: (THE BLUE BALL) (THE SMALL BLUE BALL)

In describing objects, it uses whatever attributes it knows about that object, which happen to be size and color, taken in that fixed order. Note that its two descriptions are not necessarily unambiguous, and in this case would be insufficient as references in an input. That is, in order to refer to the first ball, an input would have to include some relation that didn't also hold for the second (which relation may in fact not exist).

INPUT 10: (IS THE BOX ON THE TABLE NEAR THE BLOCK)

MiliPS's scene is sufficient to determine that after "box" the question is about a particular object, the only box in the scene. The relation "on the table" is already true of the box, so it is redundant; if the question ended after "table", MiliPS would answer "yes". MiliPS notes the redundancy and continues on, willing to abandon that answer if something negative comes along. The second relation, "near the block", is in fact inconsistent with both preceding objects, i.e., it can't be referring to either the table or the box. Inconsistency can mean that the system has definite information to the contrary, or it can mean, as in this case, that no information exists one way or the other.

REPLY 10: (NO INFORMATION ON RELATION NEAR)

It really means "on the relation near between those two objects". Note that it can do no deduction on other information that it has about the objects. For instance, it might reasonably deduce that nearness held if the block were in the box.

Once again, some declarative inputs will be skipped, to get to a sentence with new features.

INPUT 22: (WHERE IS THE BALL IN THE BOX ON THE RED FLOOR THAT IS RED)

"The ball" is ambiguous to start with, as is "the box". A unique box is determined because the floor is unique as described. When the floor is found, the system knows that there is an unused relation, "on", and backs up in a list of the current objects to resolve the box ambiguity. The same process applies to the "in", but the ball remains ambiguous. The scan through the sentence continues, and "that is red" is found to be redundant with respect to the floor (the program only checks semantic redundance, not the superficial redundance that "red" has already been used to describe the floor). The redundance leads the program to look back in the list of current objects for something that redness can apply to, and finds the main subject, the ball. The end of the sentence is reached, so a reply is constructed.

REPLY 22: (THE LARGE GREEN BALL IS NEAR IT) (THE SMALL RED BALL IS IN THE UN-RED BOX)

A "where" sentence prompts MiliPS to give the relations that an object has with others, and also the relations that other objects have with it. In the first reply above, "it" refers to the small red ball (the program doesn't keep track of the proper order of its replies, though it easily could). The "un- red box" is one that MiliPS has only been informed of as being not red. Making the reply use a subordinate clause was not considered important enough to warrant the further necessary Ps, so the "un-" form was adopted.

A final query exercises the ability to extract questions and use relations that are separated from the objects to which they refer.

INPUT 25: (IS THE BALL NEAR THE GREEN BALL IN THE BOX THAT IS NOT ON THE RED TABLE BLACK)

Here the box is not disambiguated until the end of the clause that follows it, and the subject ball is not disambiguated until the box is. The "in the box" relation restricts the subject ball, and "near the green ball" stands by itself and also restricts the subject ball. (It was somewhat troublesome to construct such a test.)

REPLY 25: (NO INFORMATION ON COLOR BLACK)

The final word in 25 expresses the question. MiliPS knows the ball is red, but cannot deduce that it is thus not black, and instead says it doesn't have positive or negative information.

The tests given to MiliPS are all expressed in a language with fairly rigid form, which can be described with a context-free grammar. Since grammar was not deemed of primary importance, a simple form with adequate power for the task at hand was preferred. The language is adequate in the sense of being able to express descriptions of objects, their relations, and their attributes, and it is sufficiently ambiguous to offer significant problems of referent determination. As others have pointed out, a strictly grammatical approach to processing natural language cannot suffice to explain or understand ordinary language use by humans, so the actual approach taken on the given grammar is one that perhaps will work in a situation where the language's apparent grammar is much more complex, but where grammar is largely disregarded and understanding is driven by semantics and pragmatics. MiliPS puts each word scanned into a word class, and simply checks the word class of the preceding word to see if the grammar would allow such an adjacency. No more global context (phrase structure or parse tree) is used in this simple error checking, except that in a couple of cases the main sentence type is used to help determine the exact word class. Almost complete reliance for detecting anomalies is thus on the semantic phase of the analysis. For more detail on the structure built to represent the input, and to verify that it isn't a parse tree, see Section B.2.

The input language for MiliPS is given in Figure B.1. There are six major types of sentences (<S>'s), which are given in the first line of the syntax. <SD> is a simple declarative sentence, <SE> tells MiliPS of the existence of a new object, <SQD> is a query about a definite object, <SQE> is a query for the existence of some object as described, <SQW> is a query that seeks an object (or all such) satisfying a description, and <SQWR> asks the relative location of an object.

The two main subcomponents of the grammar are object descriptors, <OBJ-DESCR>, and predicates or relations of objects, <REL-PRED>. "Predicates" are attributes inherent in an object, while "relations" place the object in the toy scene, giving adjacency, containment, etc. relations. A glance at the last few lines of the syntax gives a good idea of the limitations of the domain of discourse.

Needless to say, this grammar is highly ambiguous, in particular with regard to the referent of a <RELPHR> or <RELCL>. The universe of discourse consists of a "scene" with five kinds of objects, which have attributes size or color, and which can be in certain relations to one another. Any object can usually be described fully using the appropriate combination of attributes and relations. Exceptions can easily be generated by describing duplicates of some objects, but these are ambiguous in this context anyway. MILISY doesn't have the property that an object with a unique description can be described in its input language (it doesn't have subordinate clauses or the ability to conjoin relational phrases). MiliPS corrects this defect, while introducing possibility for ambiguity.

Ambiguities are resolved in a "natural" way. A phrase applies to the object immediately preceding it, unless it is inconsistent with it, in which case it applies to the

```
<S>                 ::= <SD> | <SE> | <SQD> | <SQE> | <SQW> | <SQWR>
<SD>                ::= <OBJ-DESCR> IS <REL-PRED>
<SE>                ::= THERE <COP> <INDEF-OBJ-DESCR>
<SQD>               ::= IS <DEF-OBJ-DESCR> <REL-PRED>
<SQE>               ::= IS THERE <INDEF-OBJ-DESCR> <REL-RELCL>
<SQW>               ::= WHAT <OPT-RELCL> <COP> <REL-PRED>
<SQWR>              ::= WHERE IS <DEF-OBJ-DESCR>
<OBJ-DESCR>         ::= <INDEF-OBJ-DESCR> | <DEF-OBJ-DESCR>
<COP>               ::= IS | IS NOT
<REL-PRED>          ::= <RELPHR> | <AV>
<REL-RELCL>         ::= <RELPHR> | <RELCL>
<OPT-RELCL>         ::= <RELCL> | empty
<INDEF-OBJ-DESCR>   ::= A <AVPHR> <N> <MOD-SEQ>
<DEF-OBJ-DESCR>     ::= THE <AVPHR> <N> <MOD-SEQ>
<MOD-SEQ>           ::= <RELPHR> <MOD-SEQ> | <RELCL> <MOD-SEQ> | empty
<AVPHR>             ::= <AV> <AVPHR> | empty
<RELPHR>            ::= <REL> <OBJ-DESCR>
<RELCL>             ::= <RELPRON> <COP> <REL-PRED>
<N>                 ::= BALL | BLOCK | BOX | FLOOR | TABLE
<AV>                ::= LARGE | MEDIUM | SMALL | RED | GREEN | BLUE | BLACK
<REL>               ::= IN | ON | NEAR | UNDER
<RELPRON>           ::= WHICH | THAT
```

Abbreviations in grammar names: S - sentence; D - declarative; E - existential; Q - query; W - what; WR - where; OBJ - object; DESCR - descriptor; REL - relation or relative; PRED - predicate; CL - clause; OPT - optional; COP - copula; DEF - definite; INDEF - indefinite; MOD - modifier; SEQ - sequence; AV - attribute-value (value of an attribute, i.e. of size or color); PHR - phrase; N - noun; PRON - pronoun;

Figure B.1 The input language for MiliPS

preceding object, and so on.● This "backup" occurs only past objects whose referents have been uniquely determined. Also, a phrase that is consistent with an already-uniquely determined object is said to be redundant, and may be used to restrict the referents of a previous object (more precisely, the most recent one that satisfies the following condition), if the phrase is consistent with it and if that previous object is not uniquely determined. Ambiguities for referents in <SQW> and <SQD> are handled somewhat differently, since an inconsistency might be the purpose of the query, that is, to determine if some property or relation holds. These will be discussed in detail below. Note that several consecutive prepositional phrases or subordinate clauses can apply to the same object, without a separating "and" where it would ordinarily occur in human communication.

The database consists of a simple record of properties and relations of objects described in input sentences. It is stored as a particular set of Working Memory predicate instances, which set is left intact across sentences. In declarative sentences, <SD> and <SE>, using the indefinite "a" determiner causes creation of new objects. No attempt is

● This is not an inconsistency in the database, which would be analogous to logical inconsistency in theorem-proving systems, but rather a disagreement between an interpretation of an input and the database.

made to keep the database consistent, and no inference is done to answer queries; only a simple lookup of the facts specified is done in this case, and also in the case of the processing of relations and properties for ambiguity resolution. In particular, negations of any sort are recognized only if explicit (following MILISY conventions here). There is no inherent reason why a more sophisticated data-base regime could not be implemented, but the focus of the current work is on certain of the language-processing aspects.

MiliPS's first reaction to an input is to scan across it, left to right, noting word classes and, near the beginning, assigning a type to the sentence. The sentence types, which correspond directly to the main grammatical classes descendent from <S>, are used in minor ways to guide the classification of words. In particular, how "a" is treated depends on sentence type: in a declarative sentence, it is indefinite, and results in creating a new object to which it then refers; in a <SQE> query, "a" really means "any", and is treated as if it were "the", which turns out to be the right way. Sentence type is used in a more significant way in treating unusual semantic occurrences, namely, inconsistencies, redundancies, unresolved ambiguities, and phrases that have no referents.

For declarative sentences, of type <SD> and <SE>, the response to the whole input is to add to the subject of the sentence the relation or attribute-value that follows the "is" or "is not". For these, it is known that at some point new (and thus inconsistent) information is to appear, so it doesn't treat it as an error. The presence of the inconsistency actually is a helpful cue to the processing, allowing it to be done bottom-up, rather than doing a more directed, top-down search for something new. If there is no inconsistency, there is either a redundancy, which is accepted without comment, or an ambiguity, which is an error.

Queries of type <SQD> and <SQE> ask definite questions, namely specific relations or attributes of a particular object. For these, inconsistency becomes a definite "no" or "no information", and can sometimes be detected before the end of the sentence is reached. Redundancy can be turned into either a positive or negative answer, depending on whether the redundancy holds with respect to the subject or with respect to a lesser object and is at the same time inconsistent with the subject. Ambiguities or null referents in these are errors.

For <SQWR>, which asks "where?", MiliPS simply outputs a list of of all the relations that pertain to the subject. No "unusual" occurrences are allowed. A sentence of type <SQW> desires ambiguities or null references, since it asks for which set of objects in the scene satisfy some description. It starts by assuming the full set of scene objects, when it recognizes "what", and as each relation or attribute-value in the sentence applies, the set is narrowed down. If the result of the restrictions is the empty set, "nothing" is answered. Otherwise, the object or objects in the set are "described" by adding the full list of known attribute-values to a corresponding noun.

There seem to be six kinds of completeness that are desirable in a system like MiliPS: completeness of reference, completeness of description, completeness of query logic, complete ability to manipulate the model, and complete symmetry of input-output behavior. Completeness of reference means that any object that is describable uniquely using the attributes and relations given, can be described in the language. MiliPS has this kind of completeness, although the particular set of relations it has could be augmented so that scenes that are presently relationally equivalent could be further distinguished. MiliPS also lacks certain kinds of reference to which humans are accustomed, such as being able to refer to the time recency of an object, as in "the third ball" or "the block mentioned before the red one". Completeness of description means the ability to describe a new object sufficiently so that it will be unique with respect to later attempts at describing it, i.e., so that it can be the unique referent of some phrase. MiliPS has this kind of completeness also - it allows descriptive relational phrases to be strung together indefinitely, e.g., in <SD> type sentences.

Completeness of query logic can best be described in terms of possible arrangements of definite and indefinite items in an abstract notation as follows: having an object x related to object y by relation R will be denoted xRy; similarly, x has a value v for attribute A is represented xAv. A query logically can have a "?" in one or more of the three positions of either the xRy or xAv triples, plus the forms xRy? and xAv? are allowed, to give a total of eight possibilities for each form of triple. For the xRy form, they are (using x and y as definite objects, and "on" as a particular typical relation): xR? (what is x on top of), x?y (how is x related to y), ?Ry (what is on y), ??y (what has any relation to y), ?R? (what is on anything), x?? (where is x), ??? (what relations do you know), and xRy? (is x on y). For the xAv form, they are (using color as a typical A, red as a typical v, and x as a typical object): xA? (what is x's color), x?v (what of x is red), ?Av (what has color red), ??v (what is red), ?A? (what has color), x?? (what are the properties of x), ??? (what does everything look like), and xAv? (does x have color red). For the present, we ignore the further complications of numerical and other forms of quantification, keeping the logic within a propositional system.

MiliPS does not have all of those forms of query completeness, but some are included in more general cases, as the following enumerates. The forms xRy? and xAv? are gotten with <SQD> or <SQE>; note that here and in most cases, if a "v" is given, the "A" is implicit, for instance, "is x red" rather than "does x have color red". Thus <SQD> and <SQE> include x?v. The <SQW> sentence type gets queries of the forms ?Ry and ?Av, and also, because of the 1-1 mapping between v's and A's, ??v. <SQWR> answers the relational variety of x??, and includes, but gives much more than is required, for x?? (for Av), xR?, xA?, and ??y. MiliPS has ??? for xAv variety, by giving it "what is" (not allowed, by the strict grammar above, but the program accepts without specific modification), and this also answers but gives extra, for ?A?. ??? for xRy and ?R? can be obtained by asking "what is" and then "where is x" for each thing that it gives as its reply; this gives a lot more information than is desired by the exact query. Thus, a user of MiliPS can find out everything about the scene, but only in sometimes cumbersome ways, and only if he or she does the computing necessary to reduce voluminous answers.

For MiliPS, completeness of manipulation involves being able to make changes to blocks configurations after they have been described. This would include being able to undo the effects of mistaken inputs, e.g., to remove a newly created object. MiliPS doesn't have manipulation capability at all. Completeness in symmetry of input-output behavior means being able to describe things in the same way that things can be recognized in inputs. This also is beyond MiliPS. It has internal representational features, such as color and size, that can't be used explicitly in inputs (e.g., "what color is the ball?"). Finally, completeness of definability and augmentation, which deal with defining new words and otherwise adding to a program's language capability, is lacking in MiliPS. The completeness scheme just presented has not been discussed or applied elsewhere, to the best of my knowledge, so at the moment it is difficult to say precisely how MiliPS compares to other systems.

B.2. The organization and components of MiliPS

MiliPS processing is driven by a left-to-right scan across an input. At each scan position, a word is given a lexical class, adjacencies are checked to insure local grammaticality, and appropriate semantic processing, in a hierarchy of several possible levels, is done. The processing is thus bottom-up, with the number of levels above the lexical level that do processing dependent on particular conditions. Each level recognizes its applicability and acts accordingly, and its output may result in fulfilling the conditions of the next higher level. At each scan point, the maximum that can be known about the intention of the input is actually known (how this is useful is discussed in Section B.3). The following paragraphs give general information about the processing and organization, filling in details on each of the levels.

The main components are represented as very abstract Ps (VAPs)● in Figure B.2. In order to define and clarify those components, we will abstractly follow through the processing of Test 2, for which a detailed trace appears in Appendix D. Test 2 is "A BLUE BALL IS ON THE TABLE". The test is started by a "scanned" signal on the left end of the input string, a marker position to the left of "A". VAP SN then acts to cause "A" to be scanned. The "scan" signal is processed by an instance of VAP GR1, which in this case notes the initial "A" as signaling a sentence of type <SD>. "A" is classified as an indefinite determiner (its "word-class"). Next an instance of GR3 fires, verifying correct grammar for the word - in this case, "A" signals a noun phrase is starting, so that the grammar check is for correctness of a noun phrase at this point. A noun phrase is considered grammatical if it is preceded by: the word "THERE" if this sentence is type <SQE>; a relation word, i.e., a preposition; a copula ("IS" or "IS NOT"); or the left end of the sentence. When the determiner is processed, initialization is done for a new noun phrase (VAP NP1). At this point nothing further can be done, and the scan resumes because of the "scanned" signal previously asserted by SN1 and stacked according to Psnlst's event order mechanism.

"BLUE" is tagged as an attribute-value word by an instance of VAP TG. This leads to the grammar check for attribute-value, which is a set of cases similar to the ones listed above for noun phrase. This particular case of attribute-value, because an indefinite determiner has preceded it, is not processed as in FR2, but is stored as a future restricter on the new scene object to be created when the noun of the phrase is scanned. The scan continues, reaching "BALL", which is tagged as a noun by an instance of TG. The grammar is all right because it is preceded by an attribute-value. Specific noun processing is now done (VAP NP3), influenced in this case by the indefinite determiner. A new object, BALL-1, is added to the scene, and the remembered attribute-value "BLUE" is added as its color.

Once again, the scan continues, on to "IS". The word is tagged as a copula, is checked for grammaticality, and its action signalled (NP2). A noun-phrase boundary necessitates checks that all referents are determined for current objects (VAP BR8), since

● See Chapter IV for a description of the VAP notation.

SN: scanned(previous) & next-position -> scan(next) & scanned(next); [4 Ps]
TG: scan & particular-word -> word-class; [22 Ps]
ER: error-at-position -> collect-input-up-to-error-for-reply; [4 Ps]
ET: interesting-event -> print-external-trace-message; [9 Ps]

GR1: scan & particular-initial-word -> word-class & sentence-type; [7 Ps]
GR2: scan & particular-word & sentence-type -> word-class; [4 Ps]
GR3: word-class & lexical-adjacency & context -> word-class-action; [27 Ps],
where word-class-action = {determiner, copula, attribute-value, predicate, noun,
new-relation-open}

NP1: determiner -> initialize-new-noun-phrase; [4 Ps]
NP2: copula -> noun-phrase-boundary; [2 Ps]
NP3: noun -> create-new-scene-object OR restrict-referents; [7 Ps]

FR1: question-word OR definite-determiner
-> setup-possibilities-from-all-scene-objects; [4 Ps]
FR2: attribute-value -> restrict-referents; [2 Ps]
FR3: restrict-referents & single-matching-possibility -> refers; [1 P]
FR4: restrict-referents -> delete-non-matching-possibilities; [8 Ps]
FR5: predicate -> check-predicate-restriction; [1 P]

BR1: refers(new) & new-relation-open -> check-relation-restriction; [2 Ps]
BR2: check-relation(or predicate)-restriction & new-object -> add-relation(predicate); [2 Ps]
BR3: check-relation(or predicate)-restriction & feasible-to-restrict
-> restrict-referents; [6 Ps]
BR4: check-relation(or predicate)-restriction & relation(predicate)-is-redundant
-> backup-redundant-relation(predicate); [2 Ps]
BR5: check-relation(or predicate)-restriction & relation(predicate)-is-inconsistent
-> backup-inconsistent-relation(predicate); [4 Ps]

BR6: backup-redundant-relation(or predicate)
& some-previous-object-ambiguous-and-feasible-to-restrict
-> restrict-referents; [10 Ps]
BR7: backup-inconsistent-relation(or predicate) & preceding-object
->check-relation(predicate)-restriction; [3 Ps]
BR8: noun-phrase-boundary
-> ensure-all-referents-found & update-current-current-object-pointers; [5 Ps]

MS: inconsistent(or redundant)-relation(or predicate) & sentence-type
->add-relation(or predicate) OR answer-question OR error; [8 Ps]
VR: sentence-boundary & sentence-type -> reply OR describe-object; [23 Ps]
DO: describe-object & attribute's & relation's -> reply; [15 Ps]

Figure B.2 VAPs for MiliPS

restricting phrases are not allowed to restrict things across copulas, except in one case determined by special sentence type (<GSQW>). Because of this completion nature of a noun-phrase boundary, the only current object that is really current is the main noun of the sentence, so BR8 also includes the action of making other nouns non-current (there are no such others in the present example; they occur, for instance, in case there are relation phrases in the sentence). If there were some definite noun for which a referent had not been determined, an error would be noted at this point, keyed by the noun-phrase boundary.

The description of the remainder of the sentence, "ON THE TABLE", will be abbreviated somewhat, hitting only the new points exemplified. The relation "ON" is noted as referring in part to the current object, which is the main noun in the sentence, and also in part to an unscanned object, so it is left open (to be caught later by VAP BR1). The determiner "THE" is definite, causing the process of referent-determination to be initialized (FR1) by collecting a set of all the scene objects as possible candidates. Then "TABLE" is scanned, noted as a noun, and used to restrict the set of referents for the current object (VAPs NP3, FR4). In this particular scene, there is only one table, so that all objects except the table are ruled out by the noun "TABLE". This triggers FR3, which leads to BR1, and now the relation ON is completed, making it (BALL-1 ON TABLE-1). This in turn triggers the check for relation restriction, and VAP BR2 is applicable as a special case of restriction, simply adding the relation to the new object BALL-1. In most cases, it really would be a restriction, since it would be the case that the preceding noun would still be ambiguous, with a set of possible referents, and the new relation would serve to narrow down those possibilities. After the new relation is added, the scan continues to the end of the sentence, and a sentence boundary is signalled. This first acts as a noun-phrase boundary (BR8), making the subject noun the only one current. It then triggers the main sentence actions according to cases of VAP VR, which in this case causes the formation of the standard reply, "OKAY".

There are several aspects of the components of MiliPS as outlined in the VAPs that have not been touched on by the above example. First, a "predicate" is recognized as an attribute-value preceded by copula, and is so tagged by the grammar check (GR3). It is further processed as a restriction similar to the restriction done when a new relation is formed as in the example above (FR5). That is, a predicate is an attribute-value that is placed after the noun that it restricts. The relative pronoun that precedes the copula (as in "which is" or "that is") is not used in this predicate detection, but its own grammar adjacencies must be correct, i.e., it must follow a noun or another predicate.

Second, the VAP MS represents what is done as a fairly high-level semantics process, namely it processes redundancies or inconsistencies as recognized by other semantic Ps according to sentence type. Some sentence types, as sketched in Section B.1, actually thrive on such anomalies. Third, the action of the BR VAPs has only been briefly touched upon, so we now turn to more detail on that.

As we mentioned at the beginning of this subsection, the semantics can be seen as a hierarchy of levels. These levels are reflected in the organization of the VAPs: the FR VAPs treat ambiguities of reference of noun phrases; the BR VAPs treat the assigning of relations and predicates to their proper objects, so that the best use of their information content is made in resolving ambiguities that couldn't be done previously by the FR's; MS is a last resort for handling inconsistencies and redundancies that can't be applied to ambiguities by the BR's; and VR and DO do the generation of replies based on the outcome of the other levels. As mentioned before, the main data structure used by the FR's to represent ambiguity is a set of possible referents for an object (noun phrase). The BR's use a structure composed of such sets: a linked list with the most recently-scanned object as the current one.

In finding a place to apply a new relation or predicate, the BR's always use the current object. If it is already unambiguously determined, an attempt is made to apply the relation or predicate to a previous object in the linked list. If the relation or predicate is redundant, a check is made before going ahead and trying to apply it to a previous object (BR6). That is, a check is made for the proper sort of unresolved ambiguity at some previous point in the list of objects. The check prevents irreparable damage being done on the basis of a feature whose resolution is not very urgent. If it is inconsistent, the application of it to some object is more urgent, so the backup to a previous object is tried regardless of what the result might be (VAP BR7). When such a backup is done, the linked list of objects is updated, making the preceding object the current one, and discarding the former current one forever (no later relations or predicates will be able to refer to it - to allow that would allow a strange sort of cross-over of reference, rather than the more ordinary nested reference, where a phrase refers to a close object, a later phrase refers to an object more towards the beginning of the input, and so on). Finally, the reader will notice that there is always a feasibility check before the actual restriction of the set of possible referents is done (VAP BR3). This is because the restriction process is irreversible, and maintaining that irreversibility seems desirable, the alternative being some kind of backtracking mechanism. If the restriction process were allowed to go unchecked, it might apply a restriction such that the entire set of possibilities would be thrown out, rather than recognizing a genuine inconsistency and acting accordingly. It seems reasonable to try to anticipate such conditions than to let them happen and then try to recover.

As support for the claim that no parse tree is formed, I now summarize the information that is kept as the scan proceeds across an input, and emphasize how that information is used to avoid referring back to the actual text after it has been seen once. The type of the sentence is kept (<SD>, <SE>, etc.), providing guidance for a few grammar decisions, but for the most part being used to make main semantic decisions. When an indefinite noun phrase is being scanned, the unused attribute-values are kept until the noun is reached, at which point they are added to it. When a relation is scanned, it is remembered until the noun phrase that follows it has been completed, at which point a full relation is formed (the noun phrase providing its second argument, in effect). The definiteness or indefiniteness of a noun phrase is incorporated into the representation and processing of the noun phrase immediately, even though the noun phrase is at that point quite incomplete. That is, the determiner sets up a group of noun-phrase anticipations. Question words and noun phrases are converted into sets of possible referents, discarding the lexical forms without further ado. For objects (representing noun phrases), the linked list records order of occurrence in the input, but objects are really semantic entities, no longer attached to lexical forms as would be the case in a traditional parse tree. This structure of semantic entities is the sole source of elements that are processed in making use of inconsistencies and redundancies. At no time does the scan back up and re-scan some portion of the input in order to try to assign to it a different interpretation, as is done in more conventional parsing programs (e.g. Winograd, 1972).

B.3. Production system and natural language task issues

This subsection discusses two independent sets of issues. The first set pertains to implementing various control and organization structures in PSs, to representational features, and to how the PS implementation compares to MILISY. The second set bears on the task and on more general processing of natural language: the use of adjacency checks instead of a full grammar, the determination of referents, and the need for a more sophisticated data base.

The main control mechanism is the left-to-right scan across an input. At each scan point, the processing is bottom-up, based on successive recognitions of specific P conditions. This leads naturally to a vertical organization, in the sense that at each point, the maximum is known: all levels (lexical, grammar, semantics, pragmatics) have a chance to react as fully as possible. This allows the surface structure of the sentence to be discarded. Such vertical organization is less likely in systems where syntax and semantics are more sharply separated, and is of course ideally suited to the recognition-driven nature of PSs. There is a potential for top-down operation, since Ps could set up anticipations that might affect future recognitions.

Ps can be grouped conceptually in modules that treat similar features of the internal representation. The modules correspond to levels in the hierarchy (lexical, grammar, etc.) and to reasonable units within those levels. Generally a module acts by firing a single P, so that a module tends to represent with Ps the cases that elaborate the knowledge in the module.

At a somewhat lower level in organization, the scan uses the Psnlst :SMPX event-stacking mechanism to maintain control. It emits at the same time both a "scan" signal and a "scanned" signal, the latter being stacked until the former is examined ("scan" enables the lexical classification Ps). When "scanned" is examined, it moves the scan pointer forward or signals an error in case the "scan" signal has not been consumed.

There is another issue with respect to the initial left-to-right scan, namely, the way that a large number of Ps have the "scan" signal as a condition element. This gives a strong top-down flavor, or at least makes the Ps look like a big subroutine, rather than having them driven on more bottom-up specific recognitions. This may have an efficiency cost, but that is less important than the inflexible subroutine style. A more accurate model of language processing by humans, and a more suitable one for PSs, might be to have the input string encoded in some way such that only one element at a time would become available to the lexical Ps. Note that this is enhanced by the vertical organization discussed above, since that organization distributes the computation roughly evenly over the words. These elements would be quite specific and would presumably have very few occurrences in LHSs of Ps. (This would also work fine as a model of lower-level processes, where parts of words (phonemes or whatever) would be recognized to form a symbol representing the whole word, or the best guess at what the whole word is.) A further alternative might be to break the lexical processing into a hierarchy, with fewer Ps responding to "scan" and lexical classes of items, and with other Ps responding to the outputs from those lowest levels.

The tests for grammatical adjacency are carried out in similar fashion for all of the classes in MiliPS: there is a set of Ps that recognize correct adjacencies, plus a single P whose condition is the negation of all of the correct conditions, which thus recognizes an *error condition.* This is quite clumsy if the grammar is extended, because a new P must be complemented by an extra condition in the error P. One alternative is to use sequenced control signals as is done for the control of the scan, where the second signal would be deleted by each correct adjacency P, but would otherwise be recognized as an error. A second alternative is to implicitly order the Ps by special case, that is, a P that is a special case of another is before the other in examination order. Then the error P could be one with a single condition, keyed to the signal that initiates the grammar check; it would always be more general than the specific adjacency tests because they would include a test on the initiating signal plus the actual adjacency conditions.

Two peculiarities of Psnlst are used to advantage in MiliPS. First, the F Ps (FR VAPs) in some cases fire "simultaneously" a number of times, both in generating possibilities for referents and in erasing those possibilities after further restrictions have been found. Without the automatic multiple-firing mechanism, some further control would be necessary to ensure iteration through all such firings. Second, the D Ps (DO VAPs) for describing objects are such that a set of objects can be described in "parallel" by having the Ps at each step fire a multiple number of times, one for each element in the set. This is similar to the multiple firing of the F Ps, except here there is a succession of such P firings by different Ps, whereas in the former case only a single P fired multiply. Here also, some explicit iteration control would otherwise be necessary. This kind of behavior is evident in those tests in Appendix C that involve describing several objects.

The primary representational issue in MiliPS is the choice of representing things as Ps or as Working Memory structures. In particular, the way MiliPS keeps the scene representation in Working Memory violates the principle that long-term items be stored as Ps. As it is, MiliPS erases its entire Working Memory between inputs, except for the instances of a few select predicates which are its database and which stay around for the duration of a conversation (e.g., for the full set of 25 inputs on which MiliPS was tested). To best represent the scene as Ps, some kind of discrimination network seems appropriate. This would necessitate radical changes to the present process of referent determination, since the present one forms a set of all objects in the scene, stored in Working Memory so easily accessible, and restricts the possibilities as more information comes in. The opposite method would be used if the scene were stored as Ps. As the input were scanned, a description would be formed, and as soon as the description became specific enough to evoke a scene object, a P would fire and supply a name to the description, thereby giving the system access to further information about it, to be confirmed or rejected by further inputs. The case of having evoked more than one such object would have to be considered, and some means of matching the objects in order to further discriminate them would have to be supplied.● It seems that having conflicts between objects with respect to partial descriptions arise in this form and be treated according to a general matching discriminator is more satisfactory than the present Working Memory database from the standpoint of adding further contextual cues to the discrimination, e.g., time of creation and scene configuration dynamics. It seems more satisfactory in part because of apparent problems in getting hold of a large set of objects in Working Memory and examining them in such a way as to find descriptions that are indistinguishable and to

● Cf. the canonization of objects in GPSR, Chapter IV.

find how partial descriptions of them might conflict. Storing them as Ps makes the conflicts fall out more naturally in the course of normal task processing, and sets forth a process whereby such conflicts are resolved incrementally. Some of the apparent difficulty with using Working Memory may be due to the nature of PS architectures or of Psnlst. Since discrimination nets are usually built to use a minimal number of tests to distinguish objects, it is likely that the P storage would use less computer memory overall, especially if there is some way of avoiding duplication of conditions in Ps by sharing the overlapping parts. The problem of how to store long-term information is of minor importance for the present study, which focuses more on natural language processing, so the present stopgap seems acceptable; other chapters of this thesis do focus on such storage problems.

Three other representational and low-level PS issues can be mentioned. Words are represented two different ways in Working Memory, as a consequence of limitations in efficient match power in Psnlst, namely limitations in the way constants are used in LHSs (see Section C.2). Also, many very similar Ps in the lexical recognition process could be reorganized into a set of Ps that simply recognize an element as a member of the set, plus a single P, keyed to membership in the set, that does the more complex actions now done in each P in the set. Augmentation would then be extension to that set rather than addition of a P. Some of the Ps in the description process (DO VAPs) could perhaps be more optimal by combining their actions into a single P with more actions and conditions. This is an instance of the general operation by which frequently-recurring P firing sequences are collapsed into a single firing that removes the necessity for intermediate communication signals, but that is more special-purpose. The specific case at hand is that two P firings are required to get a size-color attribute-value description constructed, where one would suffice. (At present, I am restricting such collapsings to Ps within the same module, but an automatic collapsing process might detect others.) Finally, the use of a near-total erasure of Working Memory between each input sentence has avoided the problem of inter-sentence confusion of data. Otherwise, special erasure Ps that would embody specific assumptions of what needs to be erased (and that would consume more run time) would be needed. The massive erasure is however unattractive from the standpoint of modelling a memory that fades over time, which is probably of concern to psychologists.

Several differences between the original MILISY and MiliPS are worth noting. MiliPS employs a <u>single</u> <u>uniform</u> <u>mechanism</u> to implement processing that was done by MILISY in two distinct phases: a syntactic parser and a set of semantic transformation rules. The use of PSs for both functions (although the functions have been radically redefined) indicates their flexibility and power over the particular special-purpose mechanisms in MILISY. MILISY constructs a phrase-structured tree representation of an input (or several, in case of ambiguity) and processes it semantically by rewrite operations capable of doing certain tests on the tree structure. It is not apparent whether its rules could be augmented to perform the semantic disambiguation that MiliPS performs, or not; the fact that MILISY might generate many possible parses before finding an appropriate one makes it more cumbersome at best. MiliPS makes significant extensions in MILISY's behavior, especially in its ability to disambiguate, to handle subordinate clauses and phrases, and to answer "where" questions. MiliPS is about five times slower than MILISY (16 seconds versus 3 to 4), but MILISY would undoubtedly worsen in its performance on the more complex MiliPS tests. MiliPS is run by a PS interpreter, and compiling the Ps is expected to more than compensate for such speed factors. MiliPS has a listing about 2 to 3 times as long as MILISY's. But both of these comparison measures are less than satisfactory because the two programs have diverged functionally.

Several issues can be raised in connection with the language task, which don't bear directly on the implementation as a PS. The local-adjacency nature of the syntactic checking in MiliPS may work only because the task is suitably restricted.● Certainly, the present language doesn't contain all the basic components that unrestricted language does, but if the abstract toy blocks world does represent a significant portion of what natural language is about (objects, their relations, their attributes) then there might be some justification for trying to extend the approach to more demanding tasks. It is hard to envision a syntax system that requires less effort to carry out, except none at all. The weak syntax checking done here is justified as being a source of redundancy, preventing the system from taking action on too little input or on input not adequately structured, avoiding the possibility of irreversible undesirable actions on its environment.●● There are alternative approaches to doing the same kind of adjacency tests, which might turn out to be more suitable for other grammars, especially larger ones. One is to have Ps that reject bad adjacencies, rather than requiring a positive approval action. Another is to have more expectations set up, mixing top-down and bottom-up, rather than the pure bottom-up here. The possibilities for the kind of word following some word may be fewer than the possibilities for word classes preceding some word, and a mixture of the forward and backward strategies might minimize the number of required tests.

With respect to the process of referent determination, the present process forms a set of possibilities as soon as it sees a determiner-function word, whereas waiting for slightly more input would allow the process to start with an initially much smaller list. For example, the phrase "the" might refer to many more objects than "the blue". This strategy seems to be quite easy to implement as an extension to the present process. (This is a consideration regardless of whether the scene is in Working Memory or stored as Ps as discussed above.) The overall conceptual structure of ambiguity, inconsistency, and redundancy developed here, with the idea of keeping a linked list of current objects, seems general and natural, and thus worth pursuing in more demanding tasks. There are some choice points within that process that are currently not necessary, but might become so later. In particular, MiliPS makes use of redundant information to restrict wherever it can, but that restriction might turn out to be invalid after more input is scanned. This possibility doesn't arise in any of the present tests, and may be very rare in general. Also, the possibility of mutual disambiguation is not considered here, though it probably is necessary in general. By this, I mean for instance that two objects that are related to each other in some way might be ambiguous unless in both cases the relationship is considered. Another kind of disambiguation that is not handled arises when an unresolved ambiguity can nevertheless be used to resolve a previous ambiguity, such as might be the case in the phrase "the block on the table", where there are several tables but only one block on any table.

● But see Hays (1964) for a scheme with similar emphasis, proposed by a theoretical linguist.

●● Pratt (1975) gives efficiency as a reason for using syntax; i.e., syntax is applied to ease some of the burden on semantics and pragmatics; such a consideration is not evident here because all of the ambiguities are among syntactically correct forms.

The specific organization of how redundancy and inconsistency are treated can probably be streamlined and made more flexible, now that the tests given to MiliPS have brought out a number of cases that were not envisioned in the original structure. For instance, having action depend on sentence type might be replaced by a more general component dependence, where components are present over a large set of sentences, i.e., where sentences can be classified more parsimoniously by using component features than by assigning each a distinct type. The present task is certainly restricted in that each lexical word can be interpreted in only one sense, whereas in general discourse, words must be disambiguated by lexical context or even more global considerations. Finally, the present system of disambiguation and referent determination assumes sentences are self-contained, for instance, with no pronouns or other (elliptical) references to phrases in immediately preceding ones. It is possible that most intra-sentence processing would stay intact in the face of that bigger demand, with only the need for "epicycles" to handle larger units of text. Certainly it is not hard to imagine that structures could be left open or with changeable default values, in the expectation that later inputs might fill them in. The present philosophy at the lower semantic level might be successful at larger levels: all input is converted to some internal form (for instance, surface structure of a string is not used after it has been passed in the scan), and any revision in initial expectations has to be done on that internal form without recourse to the raw external form. That is, a faithful internal representation should be amenable to mapping or restructuring in emergency situations. A form of such mapping is exemplified in the flexible way that MiliPS resolves inconsistencies using only its semantic representation.

The database inferencing capabilities in MiliPS have been intentionally kept very weak, partially because they were weak in MILISY and partially because of the emphasis on other aspects. Class exclusions on values of attributes, and relations between relations are not used. For instance, knowing an object is red doesn't give the system the ability to use that it isn't blue - "not blue" is only known if there is explicit information. The set of relations between objects might just as well be nonsense syllables, since they don't interact and are not intended to be adequate in terms of representing all spatial properties.

C. Details on MiliPS

C.1. The tasks given to MiliPS

The entire list of sentences given to MiliPS is given in Appendix C. Included is the input text, a program trace that tells major events in processing the text, and the state of the database portion of the Working Memory, from which it can be deduced what the lasting effects of the text were. In this subsection, we first examine the program trace to make that appendix comprehensible. Then we point out other appendices that the reader might find to be of interest. Finally, the full set of sentences is described briefly in terms of what features are illustrated by various subsets of sentences.

```
ISA (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (TABLE-1 COLOR RED POS)
HASREL (BLOCK-1 ON TABLE-1 POS)

2 INPUT TEXT IS " A BLUE BALL IS ON THE TABLE "
ADDING COLOR BLUE (POS) TO BALL-1
ADDING BALL BALL-1
OBJ-2 REFERS TABLE-1
ADDING BALL-1 ON TABLE-1 (POS)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BLOCK-1 ON TABLE-1 POS)
```

Figure C.1 Program trace and database for Input 2

Figure C.1 gives a segment of Appendix C. First, a display of the database is given. From it, we see that there are two objects, indicated by ISA, namely, BLOCK-1, a block, and TABLE-1, a table. The attributes of BLOCK-1 are color green and size large, given by HASAV, and similarly the table has color red. The next line, HASREL, tells that BLOCK-1 is on TABLE-1.

The next segment in the figure gives the trace that the program emits as it scans the sentence. The first two trace lines, starting with "ADDING" show what the program does when it scans the phrase "A BLUE BALL", namely, it creates an object BALL-1 (the second ADDING) and makes its color blue (the first ADDING). The next event of note happens when it gets to "TABLE", which it knows refers to TABLE-1, the third program trace line. After that, it finishes up processing the "ON", which was left hanging until the object following it was scanned. It notes that it adds the relation (BLOCK-1 ON TABLE-1) with the last ADDING line. Finally, its standard reply is made.

The database after the run is given, showing that it has added an instance to each of ISA, HASAV, and HASREL.

```
5 INPUT TEXT IS " THE BALL ON THE BLOCK IS SMALL "
OBJ-1 AMBIG B2-1 BALL-1 BALL-2
OBJ-2 REFERS BLOCK-1
RELRESTR OBJ-1 B2-1 ON BLOCK-1 POS
OBJ-1 REFERS BALL-2
PREDINCON OBJ-1 S7-1 SIZE SMALL POS
ADDING SIZE SMALL (POS) TO BALL-2
REPLY ((OKAY))
```

Figure C.2 Program trace for Input 5

Figure C.2 gives the program trace only, for a more complicated example, to show a few other features of what the program emits. The first line after the input text shows the status as of the second word, which has been tagged internally as B2-1 (decoded: the second word, which starts with B, the first token for such a word). The phrase "THE BALL" has also been named OBJ-1, and the main point of the message is that OBJ-1 is ambiguous, referring at least to BALL-1 and BALL-2 (in this case, those are the only referent possibilities, but in general, more would exist, with the same message printed). Continuing, the next trace message says that OBJ-2, the name given to the second noun phrase "THE BLOCK", has a unique referent, BLOCK-1. This means that the ON relation left hanging can be completed, noted by the "RELRESTR" line. After the restriction has been done, the ambiguity for OBJ-1 has been resolved, making it refer to BALL-2. The scan continues, reaching the predicate "SMALL". It notes that this is inconsistent with the subject BALL-2 (referred to as OBJ-1), in the line starting with "PREDINCON". In that line, S7-1 refers to the seventh word in the text string, which starts with S, namely "SMALL". Since this is a declarative sentence, the inconsistency is taken in stride, that is, it is added to the subject as a new attribute-value, signalled by the ADDING line.

Appendix D gives a rather complete trace of the behavior of the PS on Input 2, including each P firing and the changes it made to the Working Memory. The reader should be able to follow it by using the description of that test given in Section B.2. At the end is a full display of the Working Memory. To understand the meanings of predicates, consult Section C.2; the program itself and a cross-reference are given in Appendix A and Appendix B. As mentioned above, Appendix C gives the program's behavior for the full set of tests. In addition, the portion after the third segment, tests 11-15, gives a summary of the control flow between groups of Ps (according to the first letter of the P name) for that test segment.

The full set of sentences is divided into five segments, for ease of debugging and presentation. The tests are given to the program via the X Ps, given at the end of Appendix A. The first segment, tests 1-5, consists entirely of declarative sentences, describing an initial scene. The second segment is four queries and one declarative sentence. The queries illustrate some of the simpler descriptive capabilities of the system. The third segment has as its main new feature the use of "NOT", both in declarative and interrogative sentences. It should be clear from these tests that the way the program encodes and uses negation is rather primitive. The last two segments are similar. They illustrate the processing of much more complex sentences, with numerous ambiguities, inconsistencies, and redundancies to be resolved.

C.2. Meanings for the predicates in MiliPS

The descriptions in this subsection are given alphabetically by predicate. The predicates for the residual database are ISA, HASAV, and HASREL. Lexical classifications start with the letters "IS". Sentence types start with "GS". See the beginning of Appendix D for a sample of how an input sentence is represented internally. The trace itself in that appendix and the display of the entire Working Memory after the program finishes on that test should provide some clues as to which predicates might be of interest.

Predicate arguments in the following descriptions are typed according to the conventions:

a	attribute: COLOR, SIZE
o	object: BALL-1, BLOCK-3, etc.
p	position in string: T1-1, B5-1, etc.
r	relation: IN, ON, UNDER, and NEAR.
s	sign: POS or NEG
t	temporary object token: OBJ-1, OBJ-2, etc.
v	value: LARGE, RED, etc.
w, x, y, z	arbitrary.

ADDAV(o,t)	add new attribute values for t to new object o. (N)●
ANSPRED(o,a,v,s)	answer a question according to the result of testing whether the predicate (a,v,s) is true of o. (V, M)
ANSPREDFIN(a,v,s)	the predicate represented by (a,v,s) is the final word of a sentence. (V, A)
ANSPREDRED(o,a,v,s)	a potential ANSPRED is redundant. (V, M)
ANSREL(o1,r,o2,s)	answer the question according to whether (o1,r,o2,s) is a true relation. (V)
ANSRELINC(o1,p,r,o2,s)	the relation (o1,r,o2,s) is inconsistent, so answer accordingly (depending on sentence type) (V, M)
ANSRELRED(o1,r,o2,s)	a potential ANSREL is redundant (V, M)
AVRESTR(t,p,a,v,s)	restrict the possibilities for t by applying the restriction that it be (a, v, s). (F, A)
COPSIGN(s)	s is the sign of the most recent copula. (R, G)
CUROBJ(t1,t2)	t1 is the current object, and t2 is the *previous current* one; t1 and t2 may be also o1 and o2 by type (A, R, N, F, B, M, V, G)
CUROBJP(t1,t2)	t1 and t2 are previous CUROBJ pairs (B, M, V, G, N)
DEFDET(p)	a definite determiner is at p (N, G)
DEFFND(t,p)	find possible referents (FINDPOSS) for t, at p (F, N)
DESCRAV(o,a,s,x)	describe o by attaching to the list x the value for o of the attribute a, if any. (D)
DESCRIBE(o)	describe o by finding and concatenating all of the (a, v, s) properties for o. (D, V)
DESCRIBED(o,a,v,s)	o has been (partially) described using (a, v, s) (D)
DESCRNX(a1,a2)	a2 follows a1 in the predetermined order of describing the attributes of an object (DESCRIBE). (D)

● Letters in parentheses after a definition are initials of P groups in which the predicate is used.

DESCRPHRASE(o,x)	x is the final output phrase describing (DESCRIBE) o. (V, D)
DETSEEN(p)	at p there is a determiner, either definite or indefinite. (A, N)
ENDMARK(p)	p marks the left or right end of the input string. (S, T, E, A, N)
EQxxx(p)	the word at p is equal to xxx. (T, G)
ERROR(p,x)	an error has occurred at p; x is a list to be added to the reply. (E, S, A, R, P, N, F, B, M, V)
ERRORS(p,x)	error scan from right to left is at p, collecting a list x. (E)
ERRREF(t,p)	for reference in case of error, t is at p. (E, B, N, G)
FINDAMBIGP(t1,p,a,v,s,t2)	link backwards by CUROBJP relations to find a place with remaining ambiguities to attach a redundant (a, v, s); t2 is where the search started, t1 is the current place in the search, and p is the location of the (a, v, s). (B)
FINDAMBIGR(t1,p,r,o,s,t2)	like FINDAMBIGP, but for a relation (r, o, s). (B)
FINDPOSS(t,o)	o is a possible referent for t. (F, B, V, M)
GSD(z)	z is a sentence of type SD, a declarative sentence. (N, M, V, G)
GSE(z)	z is a sentence of type SE, declarative starting with "there". (M, V, G)
GSQD(z)	z is a sentence of type QD, the question form of a D type of declarative (GSD). (A, M, G)
GSQE(z)	z is a sentence of type SQE, the question form of the E type of declarative (GSE). (G, N, F, M, V)
GSQW(z)	z is a sentence of type SQW, the question form starting with "what". (N, F, B, M, V, G)
GSQWR(z)	z is a sentence of type SQWR, a question starting with "where". (M, V, G)
GTYPED(z)	z has been typed according to GSD, GSE, etc. (G)
HASAV(o,a,v,s)	o has value v for attribute a, sign s. (E, F, B, V, D, M, N)
HASREL(o1,r,o2,s)	o1 has the relation r to o2, sign s. (E, F, B, V, M)
HASRELN(t,r,s)	t has the relation r, sign s, to some object yet to be seen in the input. (B, R)
INDEFDET(p)	an indefinite determiner is at p. (N, G)
ISA(o,w)	o is an object of the class w. (E, F, D, N)
ISAV(p,a,v,s)	the attribute value (a, v, s) at p checks out grammatically; continue to process it as such. (A, N, F)
ISAVW(p,a,v)	the word at p is an attribute value (a,v); this signals the need for a grammar check. (A, T)
ISCOP(p,s)	the word at p is a copula, sign s. (G, A, R, N, T)
ISDEF(t)	t is known to be a definite object by its determiner. (A, N)
ISINDEF(t)	t is modified by an indefinite determiner. (A, N)
ISNOUN(p,w)	the noun at p, word w, is grammatically all right; initiate further processing on it. (A, R, P, N, G)
ISNOUNW(p,w)	the word at p is a noun, w; this signals the need for a grammar check. (G, N, T)
ISPRED(p)	the AV at p (see ISAV) is a predicate, which means it follows a copula. (A, R, P, F)
ISREL(p,w)	the relation word w at p is all right grammatically; continue to process it. (R, N)
ISRELPRON(p)	the relative pronoun at p is grammatically all right; initiate the normal processing for it. (P, N)
ISRELPRONW(p)	the word at p is a relative pronoun; proceed by checking whether it is grammatically all right. (P, T)
ISRELW(p,r)	the word at p is r; this signals the need for a grammar check. (R, T)
LEFTOF(p1,p2)	p1 is to the left of p2 in the input string. (S, T, E, G, A, R, P, N)
MAKISA(p,w,t1,t2)	make t1 at p into an ISA; its word is w, the previous object is t2. (N)
NEWAV(t,a,v,s)	record (a, v, s) so it can be attached to the actual object that t represents, when it becomes determined. (N, A)
NEWOBJ(o)	o is a new object (new ISA). (F, B, N)
NPBOUND(p)	a noun-phrase boundary is at p. (B, S, N)
NPBOUNDL(p)	delete the NPBOUND signal for p. (B, N)
NPGCHK(p)	check that it is grammatically correct to start a noun phrase at p. (N)
NRESTR(t,p,w)	restrict the possibilities for t at p to be nouns of class w. (F, N)
NULLREF(t,p)	the set of referents for t at p is empty. (F, V)
OCHK(t,p)	check if the possible referents for t have been restricted to a unique or null set. (F)
OLDAV(p)	the AV at p is old, ISAV has been responded to. (A, F)

OLDREF(t)	the REFERS for t has been examined. (B)
OLDREL(p)	the relation at p has been processed; ISREL has been responded to. (R)
PREDINCON(t,p,a,v,s)	the predicate (a, v, s) is inconsistent with t at p. (B, M, E)
PREDINCON T(t,p,a,v,s)	print a trace for and assert the corresponding PREDINCON. (E, B)
PREDREDUN(t,p,a,v,s)	the predicate (a, v, s) is redundant for t at p (B, M, E)
PREDREDUN T(t,p,a,v,s)	print a trace for and assert the corresponding PREDREDUN. (E, B)
PREDRESTR(t,p,a,v,s)	restrict the possible referents for t at p according to whether (a, v, s) is true. (F, E)
PREDRESTR T(t,p,a,v,s)	print a trace for and assert the corresponding PREDRESTR. (E, B)
PREDRESTRCHK(t,p,a,v,s)	check whether the corresponding PREDRESTR should be applied. (B, F)
QNOUN(p)	the noun at p is a question noun. (G, T)
QWFIND(t,p)	find possible referents (FINDPOSS) for t, at p. (F, G)
QWRDESCR2(o)	initiate the second step in the reply generation process for QWR sentences (see GSQWR and QWRREPLY2). (V)
QWREPLY(o)	use the results of the DESCRIBE process to make a reply for a QW sentence (see GSQW). (V)
QWRPHRASE1(o,x,w)	the current phrase in building the first part of the QWR answer (see QWRREPLY1) for object o is x, with word w used to separate further additions to x from the present x. (D)
QWRPHRASE2(o,x,w)	like QWRPHRASE1, but for the second part of the QWR answer (QWRREPLY2). (D)
QWRREPLY1(o1,r,o2,s)	include (r, o2, s) in the first kind of reply for a QWR sentence; the first kind gives relations of the main object o1 to other objects. (D, V)
QWRREPLY2(o1,o2,r,s)	include (r, o1, s) for o2 in the second kind of reply for a QWR sentence; the second kind gives relations of other objects o2 to the main object o1. (D, V)
QWRREPLY3(o)	generate the third kind of reply for a QWR sentence, which covers the case where o has no relations to other objects. (D, V)
REFERS(t,o)	t refers to o; t may also be of type o. (F, B, M, V, N)
RELINCON(t,p,r,o,s)	the relation (r, o, s) is inconsistent with t at p. (B, M, E)
RELINCON T(t,p,r,o,s)	print a trace for and assert the corresponding RELINCON. (E, B)
RELREDUN(t,p,r,o,s)	the relation (r, o, s) is redundant for t at p (B, M, E)
RELREDUN T(t,p,r,o,s)	print a trace for and assert the corresponding RELREDUN. (E, B)
RELRESTR(t,p,r,o,s)	restrict the possible referents for t at p according to the relation (r, o, s). (F, E)
RELRESTR T(t,p,r,o,s)	print a trace for and assert the corresponding RELRESTR. (E, B)
RELRESTRCHK(t,p,r,o,s)	check whether the corresponding RELRESTR should be applied. (B)
REPLY(x)	x is a list of words constituting an external reply. (V, E, D)
SCAN(p)	the scan is positioned at p (S, T, G)
SCANFIN(p)	the scan is finished at p. (S, V)
SENTBOUND(z)	the sentence boundary has been reached for sentence z. (V, S)
SENTENCE(z)	z is the current input sentence. (S, G, N, F, B, M)
TEXT(x)	x is the list of words in the input string. (S)
TRACING(x)	an indicator that a program trace is being printed; x is a dummy. (S, E, F)
WORDEQ(p,x)	the word at p is equal to x. (T, G, E, N)

D. Overview of WBlox

WBlox is a PS that solves blocks manipulation problems, taking commands from an augmentation of MiliPS and performing actions on the scene in order to fulfill the commands. This section and the next give an overview of the WBlox part of the system and then more details, respectively. Section D.1 presents a few examples of the problems solved by the system. Section D.2 sketches the changes made to MiliPS in order to handle the expanded task domain. Section D.3 discusses the goal-subgoal mechanism used to solve problems, and describes the way backtracking works, allowing choices to be tried, undone, revised, and tried again. Section D.4 through Section D.6 discuss issues with respect to the particular PS implementation, and with respect to implementation-independent features of the task domain that were elucidated by the present work. Section D.7 compares PSs with the original Micro-Planner implementation.

D.1. A few examples of WBlox tasks

WBlox starts with a toy blocks scene identical to that used by Winograd (1972), namely, a tabletop with a box and a variety of rectangular blocks and rectangular-based pyramids. The test sentences given to the MiliPS/WBlox system were designed to test the blocks problem-solving capabilities and exercise as many of the Ps as possible. This contrasts with Winograd's apparent preference for exercising only the natural language capabilities (though not necessarily exhausting all of them) and only using those parts of the blocks program that were evoked as a result of that. Thus what is presented here and more fully in the next section and Appendix H is a more complete demonstration of the blocks problem-solver designed by Winograd than was given by him.

The first input sentence is a simple command to put one object on another.

INPUT 1: (PUT THE SMALL RED BLOCK ON THE BLUE BLOCK)

The MiliPS part of the whole system recognizes that the small red block is not already on the blue block, i.e., that there is a serious inconsistency in the sentence. Because it involves a relation that can be associated with the PUT command, that inconsistency becomes the intent of the sentence, and is given to the problem-solving part of the system. In the initial scene, the small red block has a pyramid on top of it, so that the first problematic part of this command is to find another place to put the pyramid. This evokes the goal to GETRIDOF the pyramid. GETRIDOF in general first searches on the table for an empty place, then looks at blocks in the scene to see if space is available there. In the present case, it has no trouble finding space on the table, and proceeds to move its hand to the pyramid, grasp it, lift it to some random location within the clear region on the table that it selected, and let go of it. Now the pyramid is out of the way, so the program looks for space on top of the blue block. The blue block is all clear, and is big enough to accommodate the red one, so the program goes through a sequence of grasping, lifting, and so on, similar to that for the pyramid, to put the block in that clear space.

REPLY 1: (1 (OKAY))

The MiliPS subsystem responds OKAY after checking that what was commanded has actually been accomplished by the WBlox PS. Outputs are tagged with integers ("1" here) in case there is a set of replies, to provide a sequencing for them.

We now skip over two inputs, one asking a question and the other commanding that a green block be put in the box.

INPUT 4: (PUT THE GREEN BLOCK ON THE BLOCK IN THE BOX)

Looking at this superficially, it is ambiguous in a couple of ways. At the command level, it appears ambiguous because the system knows two ways to PUT, namely IN or ON, so that the input may be requesting a PUT . . . IN or PUT . . . ON action. This ambiguity is resolved by normal processing of the sentence: the IN phrase is needed to resolve the reference to "THE BLOCK", so that only ON remains as a candidate for the main command action. The superiority of the bottom-up approach over a top-down one is evident here, and the difference between the two can be accentuated further by adding more relations. The second ambiguity is presented by "THE GREEN BLOCK". There are two green blocks in the scene, but fortunately, both are referred to in this sentence: one is in the box, so it is the second block, which forces the ambiguity of the first one to be resolved in favor of the other one. This other green block is not on the first one, the one in the box, so that the inconsistency is taken as the intention of the command, and the WBlox part of the system can work on the specific problem posed. This problem is solved directly by moves similar to those used in the first INPUT above, since no other objects are in interfering locations. The program's reply is the same as in the preceding example.

For the next example, we skip a few inputs that had no effects of concern to us at present.

INPUT 12: (PUT A SMALL PYRAMID AND A SMALL PYRAMID AND A GREEN BLOCK AND THE SMALL RED BLOCK ON THE LARGE RED BLOCK)

Several things of note occur in the input. The use of "A" in a command causes the system to choose from among a set of existing objects that match the given description, rather than creating a new object as was the case in MiliPS alone. In fact, in this case it chooses two pyramids, taking care to make the choices distinct. The use of "AND" means that all conjoined objects are the main ones for the command, that is, the command works with a set of objects. The command is to put the set on the large red block, since the final phrase, starting with "ON", is inconsistent with the scene.

From the point of view of the problem-solving system, this command presents difficulties because all of the specified objects will not fit on the large red block unless some of them are piled on top of each other in some way. WBlox does not recognize ahead of time that the area isn't sufficient, but rather, attempts to put them on, trying a couple of variations in arrangement (which exhausts the possibilities in this case), before deciding to try the necessary packing operation. When working with a set of objects, WBlox tries to place the largest first, then the next-largest, etc. In this case, after placing three of the four objects, the space is filled, so it backs up and tries to put the third object in a different location. This fails because the third object filled up the only available space. It then backs up further and tries to put the second object in a different location. Now the second and third objects used up a rectangular region on the large red block, each filling up half of it, and the program always tries to pack objects closely together when it is putting a set of them somewhere, so that there is really no alternative place to put the second object either - packing implies using the lower left-hand corner of the region. (The program doesn't reason in this way, exactly, but tries to locate space and finds only the point already seen.) So it backs up to the first object, and can find no alternative place for it either, for similar reasons. Thus it has backed up to its starting place, and now it pursues an alternative strategy, called the PACK strategy, which says place an object, then try to put one other object on top of it, then place the next object, and so on. It puts the first object on the large red block, then puts the second object, a pyramid, onto the first object, then puts the third object onto the large red block, and the fourth on top of the third.

REPLY 12: (1 (FAILED TO PUT PYRAMID-3 ON)) (1 (FAILED TO PUT PYRAMID-1 ON))
(2 (OKAY))

The program replies that the two pyramids aren't strictly on the large red block as it had expected, and then says OKAY anyway, because some of the things it expected were fulfilled. (The first two replies are tagged identically because they were noticed "simultaneously".) The two pyramids were in fact placed on the two blocks that were placed on the large red block (pyramids being preferred by PACK for placement on top of just-placed blocks, since nothing can be put on a pyramid).

This time inputs not shown have had the system put some more things in the box, and had it add some new black blocks to the scene. It has just picked up one of the black blocks.

INPUT 18: (PUT IT IN THE BOX)

"IT" always refers to the object in the hand of the model, by convention. There is no trouble understanding the input, but severe problems in carrying it out. The program fails to find enough clear space in the box to put the block that it's holding, so it tries a drastic strategy: clearing out all the things in the box, and putting them back in in PACK mode, placing them all as closely together as possible. As above, the PACK operation includes putting every other object on top of one just placed rather than on the box proper. It succeeds, after about 65 subgoals and 70 primitive grasp, lift, and let-go actions (about ten times more than required for INPUT 1 above). The program responds simply OKAY as above.

The final example we consider here consists of building a stack of objects.

INPUT 19: (STACK UP A LARGE RED BLOCK AND A SMALL BLOCK AND IT AND A SMALL PYRAMID AND A BLACK BLOCK AND A LARGE GREEN BLOCK AND A SMALL PYRAMID)

In stacking up a set of objects, the program first chooses the largest block as the base of the stack and places it on the table. As its next step, which is repeated until all the blocks have been placed, it selects the largest block that hasn't been placed and puts it on the top of the stack (the block in the set of things to be stacked that has nothing on top of it). In this step, if the largest block that hasn't been placed is too big, it is left out, and the next one selected instead. Also, if there are two or more blocks that are the next-largest, and if one of them is already in the right place, it is left there and the process continues to the next (the program also notices if the base of the stack is already on the table when it starts). After all blocks are placed, the program selects the biggest pyramid from the set that will fit, if any, and places it. Any other pyramids must be left out.

REPLY 19: (1 (LEFT OUT PYRAMID-3))

The program checks for completion of the command by checking an internal representational set that records stacks of objects. This stack record is kept for all object movements: whenever one object is put on a block (table and box are excluded as stack members, by this definition) it becomes a member of the block's stack, or if the block wasn't in a stack, a new stack is created with both objects in it. For this reply, the program noted that one of the pyramids is not in the same stack as all the other objects that it was to stack up. This is right, because the command was not completely fulfillable, given that pyramids can't support other objects. MiliPS could in principle recognize such ill-formed commands, but it doesn't.

D.2. Changes to MiliPS for the WBlox task

Appendix E gives the portions of MiliPS that changed in converting it to translate the external language into inputs for WBlox. This subsection describes the changes, following roughly the order of their appearance in that appendix. Most of the changes, 70%, were additions of Ps, and the rest were minor changes to existing Ps, usually changing one condition or action element. No Ps were deleted. There are three main kinds of changes: lexical and grammatical changes, which are rather minor; changes to how relations are handled, adding two new varieties of relations, indirect ones and computable ones; and changes to main sentence semantics in order to interface to the blocks problem-solving Ps. After describing the changes, the varieties of blocks commands are described, along with details on main sentence semantics for them. Finally, the changes in internal representation of the scene are sketched.

In the tagging Ps (T Ps) are all of the changes that effect modifications to the acceptable language. The system now knows about PYRAMID where it used to treat BALL. To make that change, only two Ps were changed, one a T and one an N, the N that handles creation of new scene objects. The word IT is recognized as a noun phrase, and is taken always to refer to the object in the model's hand. This requires only a single P, which does all the actions necessary to make the system believe that a noun phrase just went by. This approach was taken as the easiest way to ensure that objects in the hand could be referred to uniquely, the problem being that such objects don't have the same relations to other objects that other objects do. It was easier than adding the code necessary to make use of phrases like "in the hand" or "that you are holding". IN and ON are now tagged as indirect relations, to be discussed below, and TO THE LEFT OF, TO THE RIGHT OF, BEHIND, IN FRONT OF, ABOVE, and BELOW are recognized as computable relations, also discussed below. The new prepositions UP and DOWN are also recognized, but they are only lexically treated as relations, and are otherwise just complementary modifiers for command words.

The G Ps have a number of changes relating to main grammar types. These changes also carry over into N Ps and B Ps, some of which are discussed here, others later. First, blocks commands are a new type of sentence, the imperative, or <SI>, called GSI internally. In such imperatives, "A" is taken as meaning a choice is to be made, as opposed to the old action of creating a new scene object. The actual choice is made by B Ps. The imperatives start with a particular set of command words, PICK, GRASP, STACK, and PUT; G Ps recognize these and assign the imperative type to the sentence at hand. At the same time, these words set up expectations of complementary modifiers, for instance, PICK expects UP somewhere, PUT may be followed by DOWN, etc. "AND" is recognized as a noun-phrase boundary and is used to conjoin only main sentence objects in imperative sentences. The grammatical-adjacency tests for noun phrase were rewritten to make control cleaner and augmentation easier - augmentation now requires only the addition of Ps, not also the addition of negated conditions in a P that recognizes bad conditions. Similar changes could have been made to other such Ps, but one illustration is sufficient, and the others didn't require modification anyway.

In the F Ps, the relation restriction process, by which relations are used to restrict possible referents, is split into two stages to handle a peculiar kind of ambiguity in imperatives. The command PUT expects some kind of inconsistency to occur, so that it can turn that into a command to be fulfilled, but this can interfere with the determination of referents when there is a relation that might be interpreted as both a valid restriction and an inconsistency. That is, a relation might be true of one possibility, while another possibility exists for which the relation is not true. Given the two distinct interpretations, the process assumes the relation is to be used as a normal restriction, but saves the other possibility as something that can be used in case no other inconsistency can be found. Test sentence 16 illustrates this kind of "backup".

The way that the new classes of relations are handled shows up in changes to F Ps and B Ps. Computable relations are the ones that depend on exact locations in the scene, for instance, IN FRONT OF (that locations are now exact is discussed below along with other representational changes). When these relations are completed, that is, have definite objects to which the relations are to be applied, a B P evokes a set of F Ps that assert temporary relations into Working Memory that represent specific computable relations. For instance, when "TO THE RIGHT OF THE LARGE PYRAMID" is scanned, assuming only one large pyramid, a computation is made to determine all objects to its right, and temporary representations of all of the resulting TORIGHTOF relations are asserted. These relations are used to restrict other referents in a way similar to ordinary relations and to indirect relations, to be discussed now.

Recall that the "check-relation-restriction" process (see Figure B.2), which is B Ps, checks to make sure a relation restriction is applicable before going ahead with it. In that process, when a relation that is tagged as indirect is encountered, Ps are evoked to compute temporary indirect relations from the specific relation that is the subject of the check. Indirect relations are the transitive closure of a relation, and are computed by the B10 Ps. For instance, given "IN THE BOX", a transitive closure is computed using ON, by asserting indirectly-IN for all objects ON objects in the box, and for all objects indirectly-IN, and so on. The relation ON is also given the same treatment, propagating indirectly-ON's. The actual referent-restricting Ps (F Ps) are augmented by a set of Ps that use these indirect relations in a way similar to the way the restrictions for normal relations were used before. The indirect relations are erased from Working Memory after each input sentence is finished (along with everything else except the representation of the scene). An alternative that would have required fewer added Ps would have been to assert normal relations and some record that certain normal-looking relations are really temporary, so that they could be explicitly erased at sentence boundaries. These temporary relations would then enter perhaps into blocks manipulation updating operations and into the process that describes the scene and its objects - it is not clear that this is desirable.

Now that there is provision for such indirect relations, any further classes of relations that are to be treated as temporary need not require further Ps to be handled properly. The present program has an example of this, in that computable relations are kept in the same form as are indirect ones, and don't require mechanisms beyond the initial assertion. Ultimately, if the scene should be represented as a more long-term entity in the Ps themselves, all Working Memory relations would be temporary, so that further decisions would have to be made as to differential treatment of types of temporary relations.

The M Ps have two types of changes, reflecting new main semantic action. The new <SI> imperative sentence class occurs in several P conditions that want to restrict the class of sentences to which they apply. The M60-M80s are specific Ps added to process <SI>-specific information and issue commands to the blocks problem-solving Ps. Within these, redundancies and inconsistencies are treated according to the new conventions required for imperatives, to be discussed further below.

The V Ps also have a couple of modifications and augmentations. There is a set of Ps that handles reply generation for imperatives, which includes checking that commands were actually carried out. Replies themselves are now numbered, so that textually identical descriptions can be distinguished, for instance the two "LARGE GREEN BLOCK"s in the reply to the sixth test sentence. The count of replies is initialized at the beginning of the scan by a T P.

There are four commands that are extracted from input sentences and issued to the WBlox Ps. The PICKUP command is obtained from sentences of the form, "PICK . . . UP . . . ", where either ". . ." may be empty in particular cases. For this form, referents of objects must be exact. The program checks that it is not already holding in the hand the main object in the sentence. This form will not take compound phrases, since the hand can only hold one thing at a time.

The PUTDOWN command is obtained from sentences of the form, "PUT . . . DOWN . . . ", where either ". . ." may be empty. As for PICKUP, referents must be exact, and further, the object referred to must be in the hand. Actually, all such forms can simply be expressed as "PUT IT DOWN".

The PUTON command comes from forms "PUT . . . ". The PUT can be matched to either ON or IN (the latter only goes with the BOX, and becomes a PUTON that is processed specially in some cases). This form may take compound main nouns. The system processes all such as a set, applying a single relation to them all. The specific relation to be applied to the main noun or nouns is obtained from an inconsistency in the sentence. At present, this is restricted to IN and ON, but in principle it should apply to any relation, with the intent of the command to make that relation true (the restriction is inherited from Winograd's program). The explicitness of inconsistency considerations here makes that kind of extension quite feasible, whereas it is not clear that such a general mechanism would arise naturally from Winograd's treatment (whatever it was in this case). If an input contains a redundancy but no inconsistency, or if it contains neither, it is a redundant command and requires no action; the program in the latter case will complain, but in the former will say OKAY.

The STACKUP command comes from sentences like the PICKUP one, with STACK instead of PICK. These forms must have compound main nouns, and the referents must be exact.

Finally, we sketch the representational changes necessitated by the addition of manipulations to the scene, done by WBlox. The primary change is that objects have specific spatial locations and sizes, according to a standard three-dimensional coordinate system. As in Winograd's system, an object can't be rotated, and is always rectangular and aligned with the coordinate axes. The location of an object is the location of its lower-left-hand corner (minimum x, y, and z values). There is now a hand in the scene, represented as a point with neither size nor attributes nor relations to other objects, except that it can be grasping or empty. All relations are now assumed to be positive (POS), where in MiliPS, distinction was made between POS and NEG. To have negatives would be to allow a certain vagueness that doesn't fit with exact locations (although ultimately it might be desirable, for a fully general system), e.g., "NOT IN THE BOX" would have an object seemingly floating freely at any location not on the box's surface. (This is, I believe, independent of whether "NOT" can be handled in inputs, which it now cannot be.) There is a new structure that is kept track of in the scene: the stack. A simple stack is just a set of objects, one on top of another up to some height. The generalized notion of stack here is that an object is in a stack whenever it is on top of an object in a stack. A stack is created whenever an object is placed on top of another that is not already in a stack (except the table and the box). Thus stacks really include tree-like structures of blocks - all blocks in such a structure are in the same stack.

D.3. The main components of WBlox

For the most part, the WBlox Ps work independently, as a submodule, of the MiliPS system. The language produces a single command or a set of instances representing a command on a set of objects, which evokes specific WBlox top-level Ps, which in turn evoke the full problem-solving system. When the problem solving is finished, the top-level goal succeeds and control falls back to some checking signals, left around when the WBlox Ps are evoked, which evoke a process that checks the results and forms a reply.

There are four top-level operators that are evoked from outside the WBlox system: PICKUP, which commands a specific block to be picked up; PUTDOWN, which commands a specific block to be put down on the table or wherever there is space available; PUTON, which commands that an object or a set of them be put on some other specified object (PUTON includes putting things in the box); and STACKUP, which commands that a set be stacked, one on top of another.

There are eight subordinate operators that are used by the top-level ones and by each other as subgoals to accomplish particular action sequences. PUTON1 puts a single object on another object; PACK puts a set of objects onto an object, under the constraint that they are to be packed as closely as possible. GETRIDOF involves finding some unused space to put an object and going through the actions that put it there. CLEAROFF uses GETRIDOF iteratively to clear everything off some object. PUT takes an object and places it at a specific location. GRASP attaches the hand to an object, sometimes necessitating a CLEAROFF so that it can do so, as well as an occasional GETRIDOF for what the hand is already holding. RAISEHAND computes a location above where the hand is, and moves it there. MAKESPACE tries to clear away just enough objects from a surface to free up space to fit a particular object.

The preceding set of operators all make use, ultimately, of a small set of primitive operators, which do the actual changes to the scene model and which do not further evoke other actions. MOVEHAND moves the hand from one location to another, doing all the necessary updating to object locations, to IN and ON relations, and to stack structures. MOVEHAND fails to do the motion if the location moved to is not clear to the extent required for the object that the hand is grasping. UNGRASP causes the hand to let go of an object it's holding. The converse of UNGRASP is to assert that the hand is grasping, an action that is a subpart of the GRASP action and not separated as a named primitive. The most complex primitive in the system is FINDSPACE, which is sometimes entered at one of its subordinate steps, LOCATESPACE. FINDSPACE scans the surface of a specific object to find an open region suitable for placing another object. It is the only primitive that fails explicitly with a signal that is then processed in specific ways by the evoking process. Further levels of primitiveness can be imagined, but they weren't implemented here or in the original system being imitated. For instance, MOVEHAND could involve computing actual trajectories for the motions, so that no collisions with other objects occur. These considerations are simply assumed to be always solvable and not touched on further here, although it is conceivable, for instance, that the trajectory computation might not be possible without further rearrangements of blocks.

Figure D.1 gives an outline of how the blocks commands interact. The components of the outline structure in the figure are the operators. Arguments for the operators are given in parentheses, and comments are given in square brackets. In form, the structure is an AND-OR graph, with connections of nodes to other nodes in the graph indicated by comments "above" and "iterates". This connection notation is modified to mean a copy of the structure with modifications, when such modifications are also given in the comment, e.g., "without MAKESPACE" is such a modificational comment. In numbered sequences, AND is implicit between steps, e.g. 1 AND 2 AND 3 under PUT. OR is given explicitly and means the step in question has alternatives, if the OR is between two steps with the same number, or it means the sequence of steps preceding the OR has the steps following it as an alternative, if sequence numbers differ directly before and after the OR. One ambiguity with this definition of OR is under PUTIN, where 1 is to be alternated with 1 AND 2 following the OR, not 1 AND 2 OR second 1 AND 2. The comment "primitive" indicates primitives in the above categorization of operators. The comment "iterates" means that the iteration is to be through the set in the immediate vicinity, until the set is exhausted. Details on how the various selections and primitives work, and on how sequencing is done in particular cases will be presented in Section E. The remainder of this subsection makes general comments on organization.

Most of the components given in Figure D.1 work within a set of conventions that make up a goal-subgoal mechanism. The top-level goals are commands from the input language via MiliPS. Subgoals arise as the components or operators encounter difficulties in being immediately applicable. Specific problems that can arise are encoded as Ps that recognize difficulties, and that then construct the appropriate subgoals. Sequencing of both the AND and OR types is by using a couple of specific goal-related signals, one of which (the predicate NEXT) specifies what to do if a subgoal succeeds (AND), and the other (the predicate NEXTF), what to do if a subgoal fails (OR). If neither NEXT nor NEXTF is given, the goal that evoked the subgoal succeeds. There is a small executive (5 Ps) that processes success and failure signals according to these conventions. The primitive operators in the system are not treated within these goal conventions because their operation is immediate, so that sequencing can be done with ad hoc evoking-process-specific signals. The same executive-avoiding mechanics are used for steps within goals that don't cause difficulties otherwise.

The justification for including the executive and goal-sequencing conventions is that

```
1 PICKUP(object)
  1 GRASP(object)
    1 GETRIDOF(object in hand)[if such exists]
      1 FINDSPACE(on table)[choicepoint = which location on table]
      OR
      1 FINDSPACE(on block)[choicepoint = block and location]
      2 PUT(object,location)
        1 GRASP(object)[above]
        2 MOVEHAND(location, offset by size of object)[primitive]
        3 UNGRASP[primitive]
    2 CLEAROFF(object)
      1 GETRIDOF(selected object on top of object)[above]
      2 CLEAROFF(object)[iterates]
      3 assert CLEARTOP[primitive]
    3 MOVEHAND(to center of top of object)[primitive]
    4 assert GRASPING[primitive]
  2 RAISEHAND()
    1 MOVEHAND(to location at maximum height above present location)[primitive]
2 PUTDOWN(object)
  1 GETRIDOF(object)[above]
3 PUTON(object1 or set of objects,object2)
  1 PUTON1(object1 or selected object from set,object2)
    1 CLEAROFF(object1)[above]
    2 FINDSPACE(for object1 on object2)[primitive; choicepoint = location]
    OR
    2 MAKESPACE(for object1 on object2)[only if PUTON is for one object]
      1 GETRIDOF(selected object on object2)
      2 FINDSPACE(for object1 on object2)
      OR
      2 repeat MAKESPACE(for object1 on object2)[above]
    3 PUT(object1,location found)[above]
  2 PUTON(remainder of set,object2)[iterates]
  OR [after all choicepoints within PUTON1 have been tried]
  1 CLEAROFF(object2)[above]
  2 PACK(set of objects,object2)[set excludes all objects on object2 before]
    1 LOCATESPACE(for selected object = object1, on object2)[primitive, choicepoint = location]
    2 PUT(object1 at location found)[above]
    3 PUTON1(another selected object on object1)[above; only if fit is possible]
    4 PACK(remainder of set,object2)[iterates]
4 PUTIN(object1 or set of objects,box)[comes from MiliPS as PUTON, step 1 here]
  1 PUTON(object1 or set,box)[above; only first 1-2 sequence, without MAKESPACE]
  OR
  1 CLEAROFF(box)[above; but first add what's already in box to set]
  2 PACK(everything now in set,box)[above]
5 STACKUP(set of objects)
  1 PUTON1(selected object,table or current top of stack being built)
  2 STACKUP[iterates]
```

Figure D.1 The components of the WBlox goal-subgoal system

in all but the simplest problem situations goals of the same type are evoked recursively, though there are intervening levels of goal structure between the recursive calls. That is, goals do not directly evoke themselves as subgoals, but most situations give rise to recursive nesting in some way. If in these nesting situations, a particular goal process relied on ad hoc signals for sequencing, there would be more than one instance of some signals, causing confusion between the two processes. Thus, goal status for separate invocations of the same goal are distinguished with an extra argument that names the goal. Also, the NEXT and NEXTF sequencing predicates contain within them inactive versions of signals that are to be asserted, so those signals are effectively hidden and can't interact with information from active goals. If the Psnlst interpreter distinguished between matches to a P on the basis of recency of data being used in a match, and fired the P only using the most recent data (saving others until they eventually become the most recent), then the goal executive mechanism would not be necessary. (This architectural variation has been seriously considered as an interesting PS alternative.) But Psnlst, given a P with any match at all that it has come to consider for recency reasons, fires all the instantiations it can find, old and new alike. The recursively-nested structure of Planner control isolates separate goal contexts effectively, although it hides them much more opaquely (making access to other contexts impossible) than is the case in the present PS implementation.

It is fruitful to briefly compare the present solution of goal-subgoal management to that found in the more general situation, namely in GPS (the General Problem Solver, a version of which is described in Chapter IV of this thesis). The present system is very specialized, with Ps that recognize specific differences, obstacles to success with a goal, and that construct and evoke specific appropriate subgoals to treat those differences. Thus a single P firing combines the workings of the GPS match and the table of connections, between differences found and operators that might reduce them. In all cases, a difference has a unique operator that is effective. Differences are local features of the scene, so that there is no need for GPS's general match, which would want to work on two different versions of the scene (actual and desired). The closest analogue in GPS would be the performance of matches to a described, abstract object, which contains only a few features of the scene that are relevant to the main goal. But with the present high degree of specialization goes a loss of flexibility in applying operators and in using methods. The operators are very specific, and are encoded to include their own fixed subgoal sequencing. The lack of general treatment of goals and methods means that the executive doesn't evaluate progress and shift problem-solving efforts accordingly. There is also no provision for recognizing infinite loops of goals. Certainly, looping in blocks problems is possible in general, but it may be that the present restricted operator structure can not give rise to loops, although it would if it persisted in a reasonable way in trying to attain a goal.●

One detail in the dynamic behavior of the system that is hinted at in Figure D.1 by the comment "choicepoint" is the management of alternative selections within operators. Winograd's original implementation made use of Planner language primitives to ensure that all such alternatives would eventually be explored, according to a strictly depth-first search organization. That is, whenever at certain goal points alternatives existed, information as to the nature of those alternatives was recorded, and if some failure occurred at some later time, the system would back up, undoing all effects in between the

● Example: if an object, A, is to be put on object B, but has object C on top of it (i.e., C is on A), and if the only available space to put C to GETRIDOF it is on the targeted space for A on B, and if the only available space to put C is back on A when the program attempts to MAKESPACE on B to put A, then there is potential infinite oscillation.

failure and the most recent goal with alternatives, and would choose another alternative on which to base forward action. PSs have no such mechanism built into the architecture, so it has been neccessary to adopt conventions for setting up necessary information so that alternatives can be explored in a similar way, and to code those explicitly wherever necessary. On analysis of the structure of the task, it was decided to designate only a very few locations in the search as such choicepoints. The reason why this required analysis is that the Planner code for the blocks problem solver makes very frequent use of the particular primitive that achieves this mechanism (THGOAL), but only a few uses of it are actually necessary to ensure proper backtracking, the others being used to provide other functions of THGOAL. Section D.7 will go into more detail on how the final search behavior differs.

The primary function of choicepoints in WBlox is to record the current state of goals with alternatives, and to record which alternatives have already been tried. The only choicepoints in WBlox involve locations where objects are placed. If there seem to be other meaningful alternatives in terms of the task, they have here been reduced to location choicepoints. Further, the only part of the system's actions that is recorded so that it can be undone in the act of backtracking, is the sequence of primitive actions performed, along with, for some goals involving a set of objects to be iterated through, a record of the state of the iteration (i.e., which things in the set have been tried). All other goal information, for instance the goal-subgoal structure and what has succeeded or failed, is irrelevant to the backtracking and is simply disregarded in backtracking. That is, for the most part when the system backtracks, it simply reverses the sequence of hand motions and grasping and ungrasping actions that it has done since the most recent choicepoint. Whenever one of the primitives is performed, it records an event time, an integer that is incremented each time such an event occurs, and when a choicepoint occurs, the current event time is associated with it so that the backtracking can reverse the right actions. Each primitive action is also responsible for asserting an element that says what its opposite is, so that the action can be undone. The action reversal goes through the same mechanism that is used in the forward direction, e.g. the MOVEHAND primitive is evoked, so that all the proper bookkeeping is done automatically (invisible to the backup controller).

Further details on the implementation of choicepoints will be given in Section E. Even though choicepoints have been fairly easy to implement, reducing backtracking to manageable proportions, the strict depth-first variety of backtracking used here and in the original program is not considered the best way to proceed, either in this task or in general. The particular position that the PS philosophy implies on this issue is discussed further in Section D.4.

D.4. Production system issues

The next three subsections consider the issues that arose in WBlox with respect to PSs, with respect to the language used to converse about blocks, and with respect to the problem-solving operators. Included in the first is a discussion of the suitability of backtracking as a method within a PS implementation, and what an alternative problem-solving structure might look like. Also included are features of control and organization, and a discussion of some time and space efficiency characteristics of the system. Then (Section D.5) we go on to consider in detail the extensions that would be necessary to bring the system up to the level of competence of Winograd's system on the natural language side. Finally (Section D.6), there is a discussion of some details of the blocks problem-solver, independent of the implementation as a PS, which suggest difficulties and possible significant improvements in its abilities.

The most important issue with respect to PSs is the suitability of the backtracking method inherited from the Planner version of the problem-solver. Backtracking implies that there is provision to ultimately try all possible variations in sequences of problem operators in attempting to solve a problem, if that should be necessary. These alternative sequences are tried in depth-first order, and in Planner there is little program control over which alternatives at any point are tried first. In the toy blocks domain, this has proved to be no strain on the control capabilities of PSs, although analysis has simplified somewhat the amount of backtracking that is really necessary, and, further, certain features of PSs as a language, to be discussed in Section D.7, remove some of the control needs that backtracking is used for in Planner programs.

Nevertheless, for this domain it seems feasible to adopt a strategy that requires no backtracking or backup of any kind. Such a system would always work forward from its present situation, adjusting to problematic situations by applying problem-solving methods that attack those problems directly, after analyzing to find the real causes of the problems. For instance, instead of doing backtracking within GETRIDOF, which searches among alternative locations for putting an object in an out-of-the-way place, problem operators could be applied to do direct blocks rearrangements to alleviate shortages of available space. In such a scheme, the history of the choices made in attempting to solve a problem becomes global, and is no longer associated with particular choicepoints in the goal structure. For instance, all operations that have been performed on an object, and in particular where it has been placed, would be available for examination by GETRIDOF in the process of finding somewhere else to put it. Such a strategy might produce plans for actions that are non-optimal in the sense that the same object is handled several times, each shifting it to a new location, but it is judged easier to analyze such plans after the main goal has been achieved, to smooth out such (rare) rough edges. I don't know of any real exploration of the consequences of such a strategy, although the approach is similar to the kind of information-gathering discussed by Newell and Simon (1972, chapter 12) in connection with human problem-solving behavior in playing chess. Such a scheme is not foreign to the constructs included in the Conniver programming language (Sussman and McDermott, 1973). A primary component of such a strategy is a fuller system for analyzing and describing what is problematic about a situation, and for linking such a description with available methods.

Further analysis of how things are tried in the present backtracking structure could improve WBlox's problem-solving ability, or at least efficiency, and perhaps eliminate or minimize the amount of backtracking necessary. WBlox includes all of the selection that was used by Winograd to improve the search behavior, with perhaps minor improvements in a few places (to be discussed in Section E). For example, it orders sets of objects so that the largest object is considered first, in placing them somewhere. But further orderings could improve the process even more, for instance, allowing GETRIDOF to always make best use of available space by using the smallest space large enough to accommodate an object. More details on where this is possible will be given in Section D.6. PSs are advantageous in this kind of improvement due to the power of selection inherent in LHSs of Ps.

AD-A118 469 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/2
PRODUCTION SYSTEMS AS A PROGRAMMING LANGUAGE FOR ARTIFICIAL INT--ETC(U)
DEC 76 M D RYCHENER F44620-73-C-0074

UNCLASSIFIED NL

2 of 2
AD
A118469

END
DATE
FILMED
9-82
DTIC

The present <u>implementation</u> <u>of</u> <u>choicepoints</u> (see Section D.3) illustrates how PSs might be applied to problem-solving situations in which backtracking is necessary, either because not enough analysis has been done to allow more intelligence to be built into the problem solver as discussed above, or because genuine choices do exist. In such a general case, the PS architecture allows several variations on the scheme, according to task demands. One is to use P Memory instead of Working Memory to record the choicepoints, to save Working Memory space (and matching overhead) and perhaps to avoid interference between similar information at different choicepoints. In recording choicepoints, there is always a choice between storing what has been tried and what remains to be tried, which in WBlox was resolved in favor of storing what has been tried. When the task requires much more of the choicepoint mechanism, namely keeping track of entire memory contexts to return to, as in Conniver (Sussman and McDermott, 1973), PSs offer at least two alternatives also. Presumably, it is not best in such cases to use Working Memory to store the alternative data contexts. Ps can be used to store entire states as RHSs or sets of RHSs, to be made current by the proper evocation into Working Memory. Ps can also be used to store update information, so that going from one state to another previosly-stored one is done by a sequence of P firings, each making incremental updates to the current Working Memory state. For both of these, some method of storing path information, or other evocation cues, must be adopted, so that states can be accessed. For this, in principle, either Ps or Working Memory could be used.

The overall <u>control</u> <u>organization</u> of components of WBlox is as a hierarchy, along the lines given in Figure D.1. The processing is directed by explicit goals in Working Memory, and intra-goal sequencing is done by specific ad hoc control signals. In terms of modules of Ps, which conceptually means Ps that share common knowledge assumptions, the entire system is divided roughly according to the first letter of Ps' names, but in the WBlox part, modules are larger than is warranted by conceptual organization: all of the higher-level goal parts are in the W module, and the primitive operators are in the Q module. But given that, it is still the case that generally, the action of a module consists of firing very few Ps (one, two, or three, usually), which perform some actions and pass control to another module's Ps. This is true of most of the modules in the MiliPS part, and is at least partially true in the WBlox part. In WBlox, on the average, one W P fires, then about three Q Ps fire, then control goes back to a W P. This is based on figures given in the control flow summary trace in Appendix H, after the first program trace segment. This supports the claim that PSs lend themselves easily to a modular organization of knowledge, and are the right level of conciseness to express incremental applications of such knowledge modules.

PSs are used to advantage to do a variety of <u>complex</u> <u>selections</u> within single LHSs. Several processes order a set of objects by size by using an LHS that performs a match on the set and selects the largest for its next action. Some of these make the selection under the constraint that the object will fit on top of some other object. (Details on which ones make such selections will appear in Section E.) The MAKESPACE process selects an object that is the smallest one large enough to accommodate another object. FINDSPACE uses single-P selections to find greatest lower bounds on a region along X and Y dimensions, and to find least upper bounds on the two dimensions. That is, given a point in a clear region, it selects the object that forms the closest boundary of the region in a particular direction. It also uses such selections to shift its attention from a point that is obstructed by an object to the nearest point on its boundary, which may adjoin on a clear region suitable for further examination. (FINDSPACE will also be discussed further below.)

All of these selections would be clearer to express if Psnlst had an additional simple match primitive (see Chapter VII). As it is, the expression of such selections is sometimes awkward and repetitious. But at a higher program-organization level, it might be better to have a selection module or goal, rather that having each problem operator do its own selections. Having the separate selection would be warranted if it were to become more complex, e.g., based on history or on considerations other than simple local ones or on interactions with other goals.

A variety of control sequencing devices are used in WBlox. Iterations in PUTON, STACK, and PACK are controlled by signals that record the processed elements in the sets that the operators are working on. Simple match conditions exclude these tried elements from being considered in the selections involved in these processes, and the signals are noted in the same way as primitive hand actions, so that backup can take them into account. FINDSPACE uses modifiable defaults in computing boundaries of a region, which means that as a first attempt at a boundary a default value is used, and then Ps may or may not fire according to conditions, to update those default values. Later Ps make use of the existing values without then having to be concerned with where they came from. Double signals for controlling steps in a process are used in several places: in FINDSPACE, in some grammar adjacency checks, and in checking the results of the whole blocks process. That is, a P evokes one step of a process and at the same time asserts a signal that at the proper time (when it pops out of the examination stack for events, :SMPX) evokes a P that asserts a signal that starts the next step of the process. This device avoids having the next step evoked prematurely from intermediate results from the preceding step. A disadvantage is that the control signal must be included in the Ps of the second process that may accidentally suffer from premature firing, or usually all of them, to avoid having to know too much in advance. In the cases at hand, this is not a serious problem, since the second step is one P or a small number of Ps.

The generation of the transitive closures of the IN and ON relations takes advantage of Psnlst's ability to fire a P on several sets of data "simultaneously". In this case, a set of Ps amounts to a breadth-first assertion of the indirect relations in the scene, since at each iteration of the set, all the existing indirect relations are extended by another link in the chain or network. This process simply continues until no more new relations are asserted, at which point control falls back to another signal and processing continues (see the B10 Ps).

Bookkeeping after hand moves is done under the control of specific signals. When the hand moves holding an object, relations that the object had are no longer correct, and new relations may now hold, so that checking is done in two distinct steps. Without specific control of these two steps, for instance, newly added relations would be deleted by the step that deletes the existing relations in preparation for any new ones. The program actually started out without specific controls, and was found defective.

As was the case for MiliPS alone, everything in the Working Memory is deleted between input sentences, except for instances of special (by convention) database predicates. This removes the need for more careful updating and erasing of unnecessary elements, preventing interference between sentences (which wouldn't necessarily occur), but is unsatisfactory in being rather arbitrary. More reasonable schemes such as having elements automatically deleted after being unused for some number of recognition cycles are recommended by this as well as other PSs implemented so far, but cannot be explored in practice within the present scope. Another ad hoc mechanism in WBlox is the PSMacro MAKEINSTL (see P W0), which converts the value bound to a variable in an LHS match to be an assertion at the top level in the RHS. This circumvents a deficiency in the Psnlst language (not allowing variables in predicate position, and not allowing matching of nested structures), but is justified in two ways: it is used sparingly, and it is very convenient in converting data that would otherwise require a set of specific Ps, one for each type of conversion done, according to the particular predicate in the assertion.

Over the 24 tasks given to WBlox, run times range from about two minutes up to about 40, with all but one actually under about 10 minutes, and with the average at 4.5 minutes. (There is good reason to believe that the 40 minute figure may be inflated by computer system characteristics at the particular time the run was made, by as much as a factor of 2, based on average run time per P firing, which is ordinarily about 1 second, but in that case close to 2.) The PS uses a total of about 48K words of Lisp cells, and one of the longer tests (19) uses about 5.5K for its dynamic Working Memory, of which about 2K is taken up by the residual database portion. Of the 48K in program, 27K is for the MiliPS part, 21 for WBlox. The full PS has 408 Ps, including 3 test Ps, of which 278 are in MiliPS and 130, in WBlox. Since the old version of MiliPS has 193 Ps, including 5 test Ps, 85 Ps were added to bring MiliPS up to handling the richer input language. Test 19 has a Working Memory of slightly over 400 instances, of which the database is about 100 items. In that test, even though the total number of items is large, no single predicate has a large number of instances, the most heavily loaded (with about 40) being UNEVENT and NEXT, which are concerned with backup information and goal sequencing, respectively, and which could easily be stored as Ps if it were necessary to reduce the size of Working Memory.

D.5. Extending the language system

There are a number of specific features that could be added to the present system, if it were desirable to bring it to the level of competence of Winograd's original system. In fact, many of the features discussed here go beyond the original, but seem to be within reach of the PS. MiliPS is much weaker than the original in its ability to generate interesting replies. MiliPS has no capabilities to answer "why" questions, which involve knowledge of the problem-solving history that has preceded the question. Some related aspects are being able to use past tenses, being able to deal with queries about actions, and being able to use relative time descriptions such as "the first thing you touched after stacking up the red blocks". MiliPS doesn't know certain verb forms that bear on relations that it has, e.g., "what does the box contain". It also needs to be able to understand some variants on relational phrases, for example, "the block that the pyramid is on", and to be able to deal with the converse of being "in" or "on", namely the support and containment concepts. MiliPS has very little in the way of treatment of pronouns or references that depend on the history of the conversation. MiliPS doesn't handle "and" in a general way, restricting its use to conjoining subjects of commands. The present language can't deal with certain aspects of the internal representation: sizes, locations, and stacks.

MiliPS lacks an ability to handle numbers, as in "stack up three blocks" or "supported by three boxes", and it can't answer "how many" queries. This involves being able to recognize plural forms of nouns, to enforce agreement between nouns and verbs, and to recognize more general uses of conjunction, which at present is limited to the main nouns of the input. MiliPS would have to be extended to handle negation, which in particular involves some extra Ps in the referent-determination process, that would restrict the set of possible referents in an opposite fashion to the present positive restrictions. This suggestion assumes that it is more reasonable and general to assume that all database attributes and relations have a positive sign, as was assumed here, rather than allowing both signs as in the original MiliPS. If general propositional logic is expressible in natural language, to process it in the present framework would require manipulation of sets of possibilities and their complements, and possibly saving partial results for use in restoring previous interpretations on the basis of new input. For example, in "on the block or to the right of the block", the first candidate relation might make the set of possibilities empty, so that the second alternative would have to be tried with the set that existed before the first phrase was seen.

MilPS is less interactive than SHRDLU, specifically lacking the ability to lay out choices in an ambiguity situation and allow the user to specify in a simple way which one was intended. It can't augment its language ability as could SHRDLU. SHRDLU was able to attach proper names, e.g. Superblock, to objects, and it could converse about a previously-unseen concept like "ownership" or a new structure of blocks like "steeple".

MilPS lacks an ability in many cases to rule out interpretations purely on the basis of semantics, as opposed to pragmatics, as was used in the original blocks system to rule out having the table try to pick up blocks, for instance. An exhaustive examination of the possibilities of occurrences of various kinds of relations in commands, namely whether a particular phrase is used as a restriction of possible ambiguity, as a redundancy, or as an inconsistency to be applied elsewhere, leads to some cases that weren't judged to be common enough to warrant attention in MiliPS, but that might be desired in a fuller system. One case contains phrases that are all inconsistent with the main noun, but that are at varying levels of specificity with respect to being turned into the command relations to be fulfilled by the system. For instance, in "put the pyramid in the box on the red block", suppose the scene contains no pyramid in the box, and that there is a red block in the box. In this example, both relations are inconsistent with the main noun, and both could thus be commands, but the second is more specific and consistent as a command with the first, and should thus be preferred. A second case involves a redundancy that might be inconsistency with the main noun, but is subsequently superceded by a real inconsistency. Thus bindings of relations to be command relations has to be tentative in some cases, with possible updating after more of the input is seen.

How feasible is it to make these extensions? Adding to the grammar of the language accepted is relatively easy, involving just adding grammatical classes and figuring out the appropriate adjancies to be checked. Eventually, under pressure from complex languages, it might be better to systematize and generalize to the extent of using some kind of case-based structure for grammar expectations, analogous to the current way that a "pick" command expects to contain an "up" somewhere. Also as structures get more complex, the variety of sentence types might be systematized so that processing depends not on those types but on classes of types or on attributes of types, e.g., sentence types in which an indefinite determiner should be taken as a choice, as in present imperatives. The plausibility of being able to extend the present system is supported by the completeness assertions in Section B.1, and also by the relatively clean system of treating things as ambiguities, redundancies, or inconsistencies. The number of Ps estimated to be required for such an extension is in the vicinity of 200-300.

D.6. Blocks problem-solving issues

The present blocks operators closely parallel Winograd's, but it is useful to discuss them with a view toward extension, and for the purpose of raising more general problem-solving issues. One feature that was discovered in the course of testing was the possibility of interference between goals. The particular instance of this phenomenon occurs in a few situations where the program finds space to put an object, then evokes a subgoal to grasp the object, and in the process of grasping it, manages to place another object in the target location. This occurs in the problem-solving connected with inputs 18.0 and 24 (five times in the latter), which will be discussed in Section E, and it occurs only within a CLEAROFF operation, which has GETRIDOF as a subgoal, which in turn evokes PUT which evokes GRASP, which may evoke another GETRIDOF to place some object that is in the way of the GRASP goal. Apparently no other locations in the goal-subgoal structure have such a combination where interference can occur. The trap is that the FINDSPACE is done before it is certain that all other objects are in a proper location for the follow-up operation. This problem was corrected accidentally by the program itself without specific modification, due to the iterative structure of CLEAROFF: it checks the existing situation on the object being cleared off each time it iterates, essentially double-checking previous attempts, and not assuming that those previous attempts were successful. MOVEHAND checks the target location for clear space for an object being grasped, and does nothing if the location is occupied. In the original program, if such a thing occurred, the failure to PUT the object in the space would have caused a failure, with backtracking to try to do something (blindly) to correct the error. Even though in the specific goal structure here the problem is not serious, it is the case in general that some provision should be made for such interfering goals, at least providing for some communication of intentions. In the particular space problem here, one solution, used by Sussman (1973, Section 4, pp. 88-90), is to esablish "ghost" objects that occupy space but can't be manipulated as ordinary objects. There is one other approach in the present case, a trivial change that rearranges the sequence of operators so that the FINDSPACE is done after the GRASP is finished, which is the subject of an experimental patch to the WBlox system, discussed in Section E.3. But the general problem of goal interference deserves further attention.

As discussed above, backtracking is considered not the best approach, especially for PSs, where it is possible to add as much guidance as desired. For the toy blocks domain in particular there are improvements that might eliminate the need for it altogether. A couple of things should be investigated as improvements along this line. Both considerations deal with the placement of objects in empty spaces, which process grows as the factorial of the number of objects to be placed, under the backtracking strategy used in the original blocks system. Several processes presently choose to work first with the largest object in the set of objects that they're working with, but the way that "largest" is determined is by taking the sum of their length and width, which is the metric used in the original. This might be improved by using area, by using the larger dimension, or by some measure dependent on context (for instance, when putting objects in a space narrow in width, width would be a more important consideration). Choosing the right largest object is important because such routines as PACK assume that using the largest object first will guarantee being able to fill the space, if any arrangement at all satisfies that goal.

The second consideration to eliminate backtracking is probably more important, namely, using available empty spaces, particularly on the table, more effectively. This assumes a more global view in FINDSPACE, which will be discussed below. One trick is to use a space for an object that is just large enough to accommodate the object, but that minimizes the extra space that is wasted because the object doesn't fill it completely. Some care must be taken here with shapes of spaces, since in the present system, spatial orientations of blocks can't be changed (for instance, they can't be rotated 90 degrees). Care is necessary because two spaces might be equivalent for one object, but for another object, only one of the spaces is right due to its shape. Another consideration is that before spaces are filled in some processes, a better idea must be obtained on what objects in the scene will ultimately have to be moved to allow the main goal to be attained. In some cases, this requires a rather exhaustive pre-examination. For instance, in STACKUP, it may be necessary to move only small objects off of blocks that are to go near the base of the stack, but later it may be necessary to get rid of a larger object that is presently on top of one of the blocks to go near the top of the stack. Along the lines of allocating space optimally, there are conceivably a number of heuristics, applicable in special situations, which could help guarantee a minimum of backtracking, for instance, taking account of specific sizes and shapes to fill odd clear regions. In some cases, it might be possible to anticipate the need for PACK, rather than trying the ordinary PUTON first, such as when a set of objects has too much area to fit on a surface without it. Note that in the present task, there are no esthetic considerations, nor are there practical constraints such as putting tall blocks toward the rear of the scene so that they're less likely to get knocked over in moving the arm around. These constraints might be applied to distinguish apparently equivalent locations under the criteria above.

Two things about choicepoints in WBlox deserve mention. First, they are not exactly the same as the ones that are logically present in the original program (by my examination of the Planner programs; it is difficult to tell exactly because the THGOAL primitive is used in many places that aren't choicepoints in the sense used here). In two places in the original, a set of objects was processed using the backtracking mechanism, rather than sorting the set by size as was used in other places in that program, and which corresponds to the selections used in WBlox. That is, an object would be picked at random, say from all those on top of some block, and if later processing based on that choice failed, backup would come back and cause another to be picked, and so on. Also, for the goal interference problem discussed above, the original would have failed some subgoal, causing backtracking, rather than letting the iterative nature of an operator do the double-checking as in WBlox. These differences will be discussed in more detail in Section D.7. The choicepoint mechanisms in WBlox are presently distributed in specific form in several places, rather than having a general mechanism used by the various operators that need choicepoints. The same approach is used to record specific primitive events that are backed up (undone) when a failure occurs. If there were a common process used by all choicepoints, perhaps some of the work now done in various places that requires things to be expressed with several Ps could be expressed more concisely, particularly things that have to do with evaluating whether to go ahead with a particular choice or whether to reject it, say, because it duplicates a previous one or because a numerical limit has been exceeded for such attempts.

The present FINDSPACE process returns the first suitable region found after a random selection from the points on a surface have been examined. At each such random point, a process applies to try to find the largest clear region surrounding the point. Although details appear below (Section E.2), it suffices here to point out that such a random basis leads to a program that is hard to debug because behavior is rarely reproduced reliably. It is based on the FINDSPACE in the original program, but in the course of development, several minor improvements have been made, and some major possibilities for further improvement are now evident. FINDSPACE could function best by searching a grid of points in the region, where the grid need not be any finer than the size of the object that is to be placed. For the smallest block in the present scene, the grid for the table would be 100 points, ranging down to less than 20 for a majority of the blocks. Most of the grid points would be rejected immediately due to being located inside an object already on the table. More would be included in regions already found, so that the actual work of examining the space around a point would probably be required for fewer than the maximum of 10 random points that are now examined. The process would then be guaranteed to find space if it existed, rather than the present arbitrary cutoff after 10 points (which are generally not in 10 distinct clear regions). The most sensible strategy would be to find the clear space once (especially for the table and the box, which usually have a lot of space and are used frequently as locations for other objects), and to keep the list of regions globally available and updated when objects are moved. Alternatively, rather than updating, a new invocation of FINDSPACE could first check grid points in regions that existed at the previous invocation.

D.7. Comparison of WBlox to the original Planner version

The two programs are apparently quite similar in behavior, although there are a few minor differences that arose to keep mechanisms within WBlox similar in design philosophy. There is one major qualification to comparisons of this sort: detailed behavior traces are not available for the original program, especially on the kinds of tests that are used here to verify that everything in the program is in good working order. Also in at least one case the program code was too obscure to attempt to duplicate its actions too closely, so an informed guess was made as to its function.

One behavior difference has to do with where choicepoints occur in the program. In the original, as mentioned before, when MOVEHAND failed because the movement caused one object to overlap the space of another, a failure resulting in backtracking occurred, whereas WBlox recovers by iterating the main goal that gave rise to the MOVEHAND command. (Actually this would apply to PUT in the orginal, which duplicated the overlap check in MOVEHAND, but not in WBlox.) The failure in the original could thus result in retrying some choicepoints before getting back to finding another place to put the object. The CLEAROFF operation in WBlox applies a selection by size to the objects on top of an object that need to be cleared off, whereas the original simply had a loop that selected at random, subject to backtracking choices. Thus WBlox has no choicepoint in CLEAROFF, where the original did. Similarly, in MAKESPACE, WBlox uses a selection by size, where the original relied on backtracking to correct any stupid choices.

The PUTON operation in the original program, when working to put a set of objects on another object, simply tried once to put the set on, in some arbitrary order, and on failure proceeded to try to PACK them on. WBlox selects items from the set by size, largest first, and when PUTON1 fails, tries to find alternative locations if possible before giving up and using PACK.

There are two differences in the hierarchical structure of the blocks operators, between the two versions. GRASP in WBlox does GETRIDOF, for an object in hand, before doing CLEAROFF of the object to be grasped; the two operations were done in the opposite order in the original. (WBlox follows Winograd's book here, which disagrees with the available Planner code, Card, et al., 1972, which was used to obtain details.) UNGRASP in WBlox includes support checks that were part of PUT in the original. UNGRASP refuses to let go of an object if it is unsupported, whereas the original would refuse to PUT it at an unsupported place. It turns out that UNGRASP never fails anyway, in WBlox, since other operators are sufficiently careful where they try to put things. As mentioned above parenthetically, WBlox has no check for object overlap in PUT, but only in MOVEHAND, whereas the original had it (redundantly) in both places. One minor difference between the two is that when the original does select objects from a set according to size, it sorts the whole set once, and uses the sorted list result thereafter, where WBlox simply selects the largest remaining object each time it examines the set.

The basic strategy in programming the present version was to take advantage of the selective power of the PS rather than to rely on a weak and inevitably stupid process such as backtracking to arrive at an appropriate sequence of actions. It is probably true that PSs are more suitable to situations where specific knowledge can be applied to help the program make appropriate selections, than to situations where the only available method is a weak exhaustive search.

Superficially, the two versions have some similarities. The lengths of the listings of the two programs are almost identical, both around 950 lines, although the PS listing looks more densely packed onto the page. The original program consisted of about 105 Planner theorems and Lisp functions, whereas WBlox has 130 Ps. But in the computer, WBlox uses 21K words, where the Planner version used 8.8K. One of the larger scenes for WBlox used about 2K words, where the original used 1.3K, but for a slightly smaller scene, so the two are similar in scene storage. A major contrast is run time, since the original ran in 5 to 20 seconds, as compared to about 60 times that for the PS. This is distorted in Winograd's favor by several problems given to WBlox that were intended to cause considerable problem-solving, perhaps a factor of 5 to 10 times more than any of the original ones. Thus the adjusted efficiency difference is within the order-of-magnitude improvement that is expected to result from efforts to compile Ps.

On a statement-by-statement basis, the main conclusion reached by comparing the contents of Planner theorems and Ps is that a Planner theorem, with several conditional accesses to its database, and with backtracking ultimately trying all the possible paths of execution through such a procedure, corresponds to several Ps, with each one representing one of the conditional steps in the Planner theorem. (To explain why the numbers above are so close, it needs to be pointed out that there are not many Planner theorems that convert to several Ps.) Figure D.2 gives a direct contrast between the two modes of expression. Alternatively, if actual conditional cases are few, a set of Ps can represent all the conditions and actions for all the possible execution paths through the theorem. For this alternative, some cases can usually be logically excluded, because some combinations of conditions, corresponding to paths, are not meaningful. Also, some of the Planner backtracking search is invisible at the surface level in P LHSs, hidden within the PS match.

```
theorem TC-Cleartop(consequent Cleartop(x));
  begin; local variable y;
    if not Support(x,?) then assert(Cleartop(x)) also succeed(theorem);
  Loop;
    if goal(Support(x, ←y)) then goal(Getridof(y),use(TC-Getridof)) also go Loop
    else assert(Cleartop(x)) also succeed(theorem);
    end;

W3: clearoff(g,x) & supports(x,y) & not supports(x,object-bigger-than-y)
      & not supports(x,object-same-as-y-and-lexically-greater-than-y)
    -> newgoal(g1) & getridof(g1,y) & next(g1,"clearoff(g,x)");
W6: clearoff(g,x) & cleartop(x) -> succeed(g);
    % cleartop is asserted automatically by MOVEHAND %
```

Figure D.2 CLEAROFF expressed in simplified form as a Planner theorem and as Ps

The Planner goal primitive, THGOAL, serves three functions. The first corresponds to a condition within an LHS, i.e., an access of Working Memory, so that a Planner user is sometimes evoking an explicit primitive where a PS user need not do so. Note that this puts failures to match the database in Planner into the backtracking mechanism, where in PSs it is simply a failure to match a P. The latter seems to have some advantages in clarity of expression, since it ties condition elements together into coherent units rather than having an unbroken string of them. The second function of THGOAL in Planner corresponds to evoking subordinate problem operators by RHS actions in Ps, except that Planner generally uses explicit references to appropriate theorems, where the selection is done by recognition in PSs (recognition of a signal or a goal). This can include iterating through a variety of methods (which is different from choicepoints within a method). The third function corresponds to setting up choicepoints in PSs. The PS expression of this is more complex than for Planner, but it has much more flexibility and selectivity. For these three functions, PSs thus provide means that are more direct, more flexible, and more explicit with regard to intent. That relatively little explicit mechanism in PSs was necessary to duplicate the problem-solving search built into the Planner language indicates that the Planner approach is not precisely suited to the domain at hand, and even lends itself to using blind search where slight additional knowledge (selectivity in making actions) can be quite effective in producing adequate problem-solving behavior.

E. Details on WBlox

This section presents enough details to give the reader a fuller picture of the inner workings of WBlox and to allow the reader to understand the corresponding complete detail in the appendices. First, a segment of program trace is explained, so that details of the program's behavior (Appendix H) can be followed. Section E.2 gives details on each of the problem operators. Section E.3 discusses the particular aspects of tasks, and describes a peculiarity of the backtracking mechanism along with an experiment that modifies the behavior to be less strange. Section E.4 gives details on WBlox's predicates, which are important for reading the actual Ps in Appendix F.

E.1. An example in more detail

Figure E.1 gives the program trace for test sentence 1. The first six lines give a trace of the processing of the input, similar to that for the old MiliPS program. The main thing to notice is that there remains an inconsistency at the end of that processing, and that it then becomes the intention of the command. The top goal for the problem-solving system is on the "STARTING" line, which says it is to put BLOCK-1 onto BLOCK-5. The part of the scene that is pertinent to this command is that on BLOCK-1 there is a small pyramid, PYRAMID-1, and that BLOCK-5 has nothing on top of it. The first action taken to achieve the PUTON goal is to establish the subgoal G-1, to CLEAROFF BLOCK-1 - objects with other things on top of them can never be moved, in this model of toy blocks. The line after the G-1 line is indented, to indicate that the goal established there is a subgoal of the previous one. Goal G-2 is to GETRIDOF PYRAMID-1, which at the start was on top of BLOCK-1.

The next five lines give the trace of FINDSPACE working. It selects several points at random on the table, to try to find space to put PYRAMID-1, finally settling on the region on the table with lower left-hand corner at point (600 0 0) (using standard X-Y-Z Cartesian coordinates) and with upper right-hand corner at (1200 600 0), as indicated by the line starting "FOUND REGION". To go through that more slowly, "REJECTING" indicates that the given point is within some object already on the table, so it can't be considered, but FINDSPACE uses that point to shift to the point on the boundary of the obstructing object that is closest to the first, and then looks for a clear region at that boundary point, as indicated by the "LOOKING AT" line (when it follows a "REJECTING" line). In this case, attention shifts from (780 721 0) to (780 600 0), where the first happened to be inside the box, and the second is on its lower boundary. Considering the boundary point doesn't help, because the clear region found according to FINDSPACE's limited capabilities is too small to fit the pyramid, as noted by the "REGION AT" line. The next attempt with a new random point on the table is successful, finding the large region with lower left-hand corner at (600 0 0). Using the FINDSPACE result, GETRIDOF establishes a new subgoal, G-3, to PUT PYRAMID-1 at a random point in that clear space.

PUT has GRASP as a subgoal, and GRASP in turn wants to CLEAROFF the pyramid before it grasps it; the CLEAROFF goal succeeds immediately, since the pyramid has nothing on top of it (the program does not make use of the fact that pyramids never have things on them). The line starting with (0) is the first primitive hand movement, which

```
1 INPUT TEXT IS " PUT THE SMALL RED BLOCK ON THE BLUE BLOCK "
OBJ-1 AMBIG S3-1 BLOCK-1 PYRAMID-1 ...
OBJ-1 AMBIG R4-1 BLOCK-1 PYRAMID-3 ...
OBJ-1 REFERS BLOCK-1
OBJ-2 AMBIG B0-1 BLOCK-5 PYRAMID-2 ...
OBJ-2 REFERS BLOCK-5
RELINCON OBJ-1 B5-1 ON BLOCK-5 POS
STARTING GT PUTON BLOCK-1 ONTO BLOCK-5
GOAL G-1 CLEAROFF BLOCK-1
. GOAL G-2 GETRIDOF PYRAMID-1
  REJECTING (700 721 0)
  LOOKING AT (700 600 0)
  REGION AT (600 600 0) TOO SMALL
  LOOKING AT (700 9 0)
  FOUND REGION (600 0 0) TO (1200 600 0)
. . GOAL G-3 PUT PYRAMID-1 (900 451 0)
. . . GOAL G-4 GRASP PYRAMID-1
. . . . GOAL G-5 CLEAROFF PYRAMID-1
          G-5 SUCCEEDS
        (0) MOVING HAND FROM (0 100 400) TO (150 150 200)
        (1) GRASPING PYRAMID-1
        G-4 SUCCEEDS
      (2) LIFTING PYRAMID-1 FROM (100 100 100) TO (900 451 0)
      TAKING PYRAMID-1 FROM STACK-3
      STACK-3 DISMANTLED
      (3) LETTING GO OF PYRAMID-1
      ADDING PYRAMID-1 ON TABLE-1 (POS)
      G-3 SUCCEEDS
    G-2 SUCCEEDS
  G-1 SUCCEEDS
FOUND REGION CLEARTOP BLOCK-5
GOAL G-6 PUT BLOCK-1 (400 640 400)
. GOAL G-7 GRASP BLOCK-1
. . GOAL G-8 CLEAROFF BLOCK-1
      G-8 SUCCEEDS
    (4) MOVING HAND FROM (950 501 100) TO (150 150 100)
    (5) GRASPING BLOCK-1
    G-7 SUCCEEDS
  (6) LIFTING BLOCK-1 FROM (100 100 0) TO (400 640 400)
  (7) LETTING GO OF BLOCK-1
  ADDING BLOCK-1 ON BLOCK-5 (POS)
  MAKING STACK STACK-4 BLOCK-1 BLOCK-5
  G-6 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))
```

Figure E.1 Program trace for WBlox input sentence 1

moves it from its starting location to the center of the top of the pyramid, which point is computed from the location of the pyramid (100 100 100) and its size, also (100 100 100). The next line, starting with (1) to indicate another primitive hand movement, shows the hand actually grasping the pyramid. The numbering of the hand movements reflects the internal bookkeeping (the actual value is called EVENTTIME) that is being done in case backtracking is required: only the hand movements and some assertions that keep track of what's been tried in connection with commands that have multiple inputs (PUTONSET, STACKUP, and PACK) are recorded in this way and subsequently undone in case backtracking occurs (the latter do not appear in the program trace, so there will appear to be gaps at times). When backtracking is going on, the program trace prints again those hand movements, but reversed to show their undoing, with the same numbers attached. That backtracking is occurring is thus evident by the descending numbers for those movements. Only a few of the tests given to WBlox require backtracking, as will be discussed in Section E.3.

After the grasping movement, the GRASP goal, G-5, succeeds, and control returns to the parent goal, the PUT goal G-3. The six lines in the trace up to "G-3 SUCCEEDS" show the completion of the PUT operation, with a hand movement lifting the pyramid to the target location, and with a further hand movement to let go of it. The other lines show the bookkeeping that is done as a side effect of the movements. First, when the pyramid is moved, it is no longer on BLOCK-1, so that the stack composed of the pyramid and the block, STACK-3, is no longer a stack. Second, when the pyramid is let go, the program notes that it is now on the surface of the table, and records that fact internally.

The remainder of the trace shows little that is new, as the program proceeds to put BLOCK-1 on top of BLOCK-5. In this case FINDSPACE doesn't need to go through the process of looking at random points because the target block is all clear. When BLOCK-1 is finally placed on BLOCK-5, a new stack is created, and both blocks are added to the stack, STACK-4. If any other blocks are added to an existing stack, i.e., are put on top of a block in an existing stack, the attendant operation consists of just noting the addition. This trace has illustrated most of the variety that the reader will encounter in looking over the program traces in Appendix H.

Other features of the material displayed in the appendices include run statistics, production-firing traces, displays of the residual Working Memory instances which compose the program's database, and diagrams of the scenes. All of these except the last should be familiar from the descriptions given of the old MiliPS program. An example of a diagram of a scene is given in Figure E.2.

The diagram shows only the horizontal plane of the scene, with the Y dimension somewhat compressed. Scattered throughout, at points approximately corresponding to actual locations of lower left-hand corners of objects, are markers for the scene objects. The object markers are systematic abbreviations of the objects' names and attributes as follows. Each marker is four characters long. The first character is the first letter of the size attribute-value for the object, if any, e.g., L for LARGE, or just the character "+". The second character is the first letter of the color attribute-value, e.g. R for RED, or "+" if it has no color. The third character is the first letter of the kind of object, e.g., B for BLOCK. The fourth character is the number of the object, i.e., the thing following the "-" in the object's name, e.g., 5 for BLOCK-5. Two exceptions to the above rules are observed: "X" is used for BOX, so as not to conflict with BLOCK, and no string is given for the table, whose location is (0 0 0). A full example is "SRP3", standing for "small red pyramid, PYRAMID-3". As to the spatial location of these four-character markers, two things need to be explained. When two objects are at the same X-Y plane location, but one is above the other (Z dimension), this is indicated by placing the marker for the higher one above

```
......................................................................
.                                                                    .
.                                                                    .
.                                                                    .
.                                                                    .
.                                                                    .
.                                                                    .
.                                                                    .
.                                                                    .
.                         ++H1                                       .
.                 LBB5  SRB1        ++X1LBP2                          .
.                                                                    .
.                                                                    .
.                                                   SGP1             .
.                                                                    .
LRB3                                                                 .
LGB4                                                                 .
.                                                                    .
.                                                                    .
.                           SRP3                                     .
.                      LGB2                                          .
......................................................................
```

Figure E.2 A display of the scene after the first command

the marker for the lower one, in the diagram. But having one marker above the other can also indicate that two objects are adjacent on the same plane, so when such ambiguities arise, the display of the database must be consulted, in particular the LOCAT predicate. Also, when two objects are too close together, i.e., would be displayed at the same place in the diagram, the second one is shifted to the right until the first open space occurs, and is placed there.

The system's behavior on Test 1 is displayed in complete detail in Appendix I, including details of each P firing and a display of Working Memory after the sentence has been processed.

E.2. Details on components

This subsection will give details on the components of WBlox that do a significant part of the problem-solving. The primary concern is to present information on the parts of the program that use selections (analogous to sorting), iterations through sets, and choicepoints. For more detail, the reader should consult the listing of the actual Ps, Appendix F, in conjunction with the information given in Section E.4. Examples of where most of the capabilities are exercised will be discussed in Section E.3.

CLEAROFF is a simple iteration of the GETRIDOF operation. CLEAROFF has two components, one to select the largest object on top of the object to be cleared off, and one to recognize success and end the iteration. The selection of the largest object is on the basis of the sum of the length and width of the object, and ties are broken arbitrarily by using lexicographic order on the objects' names. The selection results in establishing a subgoal to GETRIDOF the selected object.

GETRIDOF makes three attempts within the WBlox choicepoint mechanism to find a suitable location on the table at which to place the object to be gotten rid of, and failing that, attempts to place it on some other object that may have enough space. Whether a location is suitable or not depends on whether the whole process backtracks to the particular choice of location or not. GETRIDOF uses FINDSPACE to locate clear spaces of the required size, and for the table has to allow the possibility that FINDSPACE will return a location that has already been tried. Such a duplication is counted as one of the three attempts because it is possible that only one suitable location on the table exists. If FINDSPACE fails to find a region on the table, three attempts are considered done, and GETRIDOF goes immediately to the consideration of other objects. GETRIDOF chooses the non-table objects on which to try to find space arbitrarily (lexicographically on the name), from the set of objects (except boxes and pyramids) that are large enough to accommodate the object to be disposed of. When all the available choices fail to survive later actions, GETRIDOF causes a backup to the previous choicepoint, if any.

FINDSPACE is driven by randomly selected points on the surface on which it is to find space. The only exception to that is when the surface is completely clear. The random point is not chosen from the entire surface, but from a surface whose upper and right-hand boundaries have been trimmed by a fraction (presently two thirds) of the size of the object to be placed (clearly most points in this edge space are unsuitable because the object if placed there would protrude over the edge of the space, but some part of the space must be included so that random points near the edge are considered). In attempting to find space, ten random points are tried, and then the procedure fails. FINDSPACE works solely with the length and width dimensions, due to a restriction on the task environment, namely that an object on top of another must have its entire bottom surface in contact with the supporting object. This restriction guarantees that the space directly above any clear region on an object is clear. A random point is first examined to determine whether it is inside some object, and if so, it is replaced by the point that is closest to the random point on the boundary of the obstructing object.

Using the given point, FINDSPACE then establishes lower boundaries on the clear region aroung the point by finding the closest object in both the X and Y dimensions independently. This suffices for the present task but is not the best imaginable procedure because the result is a point that may be adjacent to clear space in one direction or the other, so that the region found might be expandable either way with possible contraction in the other dimension. A more exact procedure would take into account interactions between the X and Y bounds rather than considering them independently. (The code for doing this in the original program was rather obscure, so I tried to imitate the best guess as to what it did.) After establishing the lower bounds, the region at the lower boundary point is examined to see if it is big enough. This is done by an easily-expressed PS pattern that tests whether any object overlaps the space defined by the point augmented by a region of the desired size. If a fit is possible, upper bounds for the region are found by again testing X and Y coordinates independently, locating the closest objects in back of and to the right of the given random point. The final augmented region is used to determine the location returned by FINDSPACE, by taking the lower left-hand corner in it, by taking a random point that will still allow the object of the desired size to be placed, or by computing the point such that the object will be centered within the space. These options are chosen according to whether the space is to be packed, is on the table, or is otherwise on a block, respectively.

PUTON can come from the MiliPS part of the system as a single assertion or as a set of similar assertions. In the former case, it is immediately converted to a PUTON1 goal. In the latter case, a set is formed of the objects to be PUTON and the goal becomes a PUTONSET goal, to put that set on the target object. Before starting, PUTONSET sets up a choicepoint, so that in case there is a failure to put on the whole set of objects (resulting in backtracking to the choicepoint), an alternative strategy can be tried, to CLEAROFF the target object and to then PACK the set of objects onto it. PUTONSET iterates over the set, establishing a PUTON1 goal for each object selected. The selection is by the size of the object and, within sets of equivalent objects by size, arbitrarily by lexicographic order on the name. Each object selected is recorded so that future selections don't use the same object. That record is subject to backup, so that there is also recorded something allowing that record to be undone, similar to the undoing of a primitive hand action. PUTONSET is also used to do the first part of a PUTIN goal, but for PUTIN the action taken when there is a failure to put all the objects on is to add objects that are already in the box to the set of objects, to CLEAROFF the box, and then to PACK the set onto the box.

PUTON1 includes no selections of objects from sets, but does involve setting up a choicepoint, recording the location of the clear region found by FINDSPACE. When backtracking occurs, PUTON1 retries FINDSPACE to see if there are any alternative locations. It will try up to three times to find locations, and then will fail. In case PUTON1 fails and is not a subgoal of the PUTONSET procedure, it evokes MAKESPACE, which tries to remove objects from the target object until space can be found. MAKESPACE takes off objects according to size, preferring the smallest object larger than the desired space, but if none of those exists, removing the largest object and iterating that removal until space does exist. PUTON1 has one other variation, namely, it checks to be sure that the target object is in fact larger than the object to be placed, and if it isn't, fails.

STACKUP, like PUTONSET, takes a set of objects (blocks or pyramids), selects from the set according to size, records the selection so that it can be undone in backing up, and uses PUTON1 as a subgoal. But in addition to the size criterion, STACKUP must first use blocks, and if any pyramids are to be stacked up, one is selected to be put on the very top of the stack. STACKUP uses a match pattern to decide where the present top of the stack is, and always checks, when it is making the selection for the next object, whether some object that hasn't been tried yet is already on top of the block at the top of the stack. If the object that is accidentally in place already is not smaller than any of the other untried objects, it is left in place and recorded as an attempt as if it had been moved to that position. If the PUTON1 operation fails for one of the blocks in the set, the process goes on anyway, and that fact is duly reported in the program's reply. Note that this strategy does not always lead to the maximal stack, since the program's size metric is based on the sum of length and width, and since a very "large" object may have such a strange shape (e.g. very long and narrow) that no further objects can be put on it. No size metric used by itself can be suitable for building stacks. A more successful procedure would have to study the specific blocks' sizes in order to avoid this difficulty.

PACK is very much like STACKUP and PUTONSET in its basic operation, except that it doesn't use PUTON1. It wants to put a set of objects on top of another object in the most space-economical way possible. It evokes FINDSPACE and records the results as choicepoints, as PUTON1 does, and tries three locations, including duplicates, before failing back to the previous choicepoint. For each block so placed, there is an additional step that attempts to put something from the set on top of that. This secondary stacking is only one layer high, and after something is placed by that step, the process returns to putting things on the original target object. In making the secondary layer, pyramids are preferred to blocks, because blocks are more valuable in making the primary layer since they can be put upon. Also, the selection for the secondary layer is based on placing the largest (by the usual metric) object that will fit. If the secondary placement attempt fails, the process continues with the basic step.

E.3. Features illustrated by the tasks

The tasks given to the MiliPS/WBlox system are divided into eight segments, each consisting of three or four tests, which were so divided to allow easy testing of the program. The tests are stored as RHSs of Ps that are evoked by user commands, displayed at the end of Appendix F. Program behavior is given in Appendix H, and there is a very detailed trace segment in Appendix I. This subsection will go through the features of each segment.

The first test in the first segment has been discussed at length in Section E.1. The second test is a query that the system answers by describing a number of objects. Some of the objects are identically described by the system, but the practice of numbering the replies allows them to be distinguished to some extent by the user. The list of objects in the reply may be surprising in that the system uses comparisons of objects' lower left-hand corners to determine whether one is to the right of another, sometimes going against standard usage. The third test is a simple command similar to the first test, involving the box instead of a block, and using one of the new computable relations ("to the right of") in specifying the object to be moved.

The second segment has four tests, 4 through 7, of a similar nature to those in the first. Test 4 shows the system successfully handling a superficially ambiguous sentence. Tests 5 and 6 are straightforward queries. Test 7 shows a command involving a compound construction for the main object of the command, namely to put two objects somewhere.

The third segment also contains four tests, 8 through 11, three of which are queries that divulge no important information. The command Test 9 was originally intended to try to make the box too full to fit in further objects, but it fails to put the program into any unusual behavior.

The fourth segment has three tests, 12 through 14. Test 12 commands the program to put four objects on top of a large block. Two of the objects are specified by the identical phrase "a small pyramid", which the program correctly interprets by making two distinct choices of objects. In the course of carrying out the command, the program is forced to do backup in the PUTONSET procedure, back to the beginning of the process. It goes forward again using PACK this time, putting the objects on in two layers, with the pyramids not directly on the target object. This extra stacking causes the reply to seem as if the process was only partially successful, when in fact it was successful within its capabilities. There is a way to put one of the pyramids directly on the target object instead of packing it in the second layer, since packing the first one into the second layer means that the remaining objects can all be put directly on the target, but the program fails to see such subtleties, and continues to put alternative objects into the second layer. Test 13 puts another block into the box, making it even more crowded, and Test 14 adds four new black blocks to the scene, making table space more scarce. Note that Test 14 adds its objects without using the language, since the language doesn't have any capabilities for describing all the necessary attributes of new objects, particularly size and location.

The fifth and sixth segments, Tests 15 through 18.5 (six tests altogether) are mostly concerned with trying to fill up the box so that the program has to resort to clearing it out and packing the contents in more carefully. Test 15, which isn't directly involved with that strategy, puts a block on a block that is already full, forcing the program to use MAKESPACE to be able to fit it on. The rest of the tests deal directly with putting things in the box. Test 16 has an interesting form of ambiguity, where the program makes one choice for the referent of a phrase and then has to "back up" and take an alternative choice when it discovers that that is necessary in order to have an inconsistency in the sentence that can be turned into a command. The program doesn't really back up though, since it duly records the alternative when it makes the first choice, so that it can easily be switched if necessary; this was discussed more fully in Section D.2. Test 18.0 achieves the goal of forcing the program to clear off the box and pack things in more carefully. Tests 18.5 and a repetition of 18.0 were included in the test sequence just in case the first presentation of 18.0 failed to do it (18.0 uses "it" to refer to what is in the hand, so that it really does something new when it is repeated). The tests were not presented interactively, but in an unconditional "batch" mode, so that 18.5 and the second 18.0 were done even though 18.0 alone would have been sufficient in the particular test run - recall that the "randomness" of FINDSPACE makes it difficult to repeat particular behavior.

The seventh and eighth segments, Tests 19 through 24, are designed to force the table to be too crowded, so that the backtracking within GETRIDOF could be demonstrated. Test 19 exercises the STACKUP procedure and stacks up a number of blocks so that they can be out of the way while the table is cluttered up with other things. The set of things to be stacked included two pyramids, which the program refused to try to do, with the proper warnings. A dump of Working Memory appears after Test 19, to illustrate the kind of information that is stored to record progress within the system of choicepoints, and to illustrate goal-sequencing information. Tests 20, 21, and 22 put objects on the table, and so does 23 except that it turned out not to be necessary in the test sequence in order to produce the backtracking behavior in Test 24.

The backtracking behavior that resulted from Test 24 is rather strange: in trying to pick up the bottom block of the big stack built by Test 19, it gets rid of almost all of the things on top of the bottom block, but then fails to get rid of a particularly large block, and thus has to back up. But the backup takes it immediately all the way back to getting rid of the top of the stack, rather than the more natural-seeming operation of first trying to get rid of the lower objects in different ways, and then working back up to the top if those don't work out. This is due not to the explicit choicepoints in the PS but to the structure of the GETRIDOF process: it finds a place to get rid of the object, then tries to grasp, which in turn triggers a GETRIDOF when the object being disposed of turns out to have something on top of it. The problem with this is that the choicepoint occurs before the subgoal is evoked so that when backtracking occurs, all of the choicepoints occur before all of the hand movements, resulting in going back to the point where the stack hasn't yet been touched as described above. The behavior exhibited on Test 24 in the eighth segment is, I believe, identical to what would have been done by the original Planner version (it wouldn't have survived in that form if it had been properly tested, I speculate). (This belief is based on "hand" simulation of Planner, and would only be contradicted if Planner's implementation of handling choicepoints is contrary to what seems to me to be the natural order of things; I could not find in the available documentation anything describing that scheme in detail - there is only vague informal description of Planner primitives' semantics). The remedy is to modify the subgoal structure of GETRIDOF, so that it does a GRASP before it does the FINDSPACE. One alternative that might be easily implemented in the PS version, but quite impossible in Planner, is to have backup return to the choicepoint with the most recent primitive hand action, as opposed to the one with the most recent creation. That is, backup would undo things between two specified choicepoints, rather than treating choicepoints as a stack and undoing things from the top only. For the purposes of demonstrating the correctness of my diagnosis, I modified WBlox (with in-core edits that aren't reflected in the main program listing) and ran Tests 22 and 24 again, labelling the reruns to be the ninth segment in Appendix H. The changes to get it to work involved interposing a GRASP subgoal in the RHSs of W11, W13, and W15, and two other modifications that might also be considered fixes of bugs in the GETRIDOF choicepoint bookkeeping, although they don't interfere with the standard GETRIDOF (because the standard version in its backup throws away all of the GETRIDOF goal structure and essentially starts from the beginning again): the NEGATE in W16 has to be (ALL,-6,-9), leaving the HASLEVEL attached to the GETRIDOF goal so that it can be retried, and an extra conjunct in the RHS of W17 is necessary, GETRIDCHOICE(K+1,G,1,02,0,0,0,0), a dummy to make GETRIDOF really act as if it has tried three times on the table when it fails to find space on it in the first attempt. The behavior exhibited in the ninth segment shows a reasonable backup order, although there are redundant GRASPs because only a minimum amount of patching was done to get the desired behavior.

E.4. Meanings of predicates for Wblox

This subsection explains the predicates that are used in the additions made to MilPS to handle inputs for WBlox, and in WBlox itself. In a few cases, old predicates have been modified slightly as noted.

Many of the new predicates in the MilPS part start with "IMP" (imperative) or "CHECK". The following are used in goal sequencing and bookkeeping: HASLEVEL, FAIL, NEXT, NEXTF, SUCCEED. To keep track of events and do backtracking, these are used: BACKUP, CHOICECOUNT, CHOICETIME, EVENTTIME, UNEVENT.

Arguments to the predicates are typed according to the following conventions:

a	attribute: COLOR, SIZE
c	set or stack
g	goal
h	hand

n	number
o	object: BALL-1, BLOCK-3, etc.
p	position in string: T1-1, B5-1, etc.
r	relation: IN, ON, UNDER, and NEAR.
s	sign: POS or NEG
sx, sy, sz	size along the three dimensions
t	temporary object token: OBJ-1, OBJ-2, etc.
v	value: LARGE, RED, etc.
w	arbitrary
x, y, z	values of the three spatial dimensions.

ADDINSET(r,o,c)	add to set c objects that are related by *r* to o. (W)●
BACKUP(n)	back up in the processing, undoing actions until the choicepoint n is reached. (W)
CHAINREL(r1,o1,r2,o2)	add HASINDRELs asserting *r1 of o1 for things that are r2 of o2*, forming the *transitive closure of r1 of o1*. (B)
CHECKFAILFIT(n,o,x1,y1,x2,y2,z,x3,y3,sx,sy,sz)	if this signal is examined, the GROWTOFIT process has failed, since it deletes this when it succeeds; failure means another iteration of FINDLOWPAIR is necessary; arguments as for FINDLOWPAIR. (Q)
CHECKPICKUP(o)	initiate the CHECKPICKUP2 check. (V, M)
CHECKPICKUP2(o)	do the actual check that the PICKUP command on o succeeded. (V)
CHECKPUTDOWN(o,x,y,z)	initiate the CHECKPUTDOWN2 check. (V, M)
CHECKPUTDOWN2(o,x,y,z)	check that o is now put down, i.e., on something, with a location different from (x, y, z). (V)
CHECKPUTON(o1,r,o2)	initiate the CHECKPUTON2 check. (V, M)
CHECKPUTON2(o1,r,o2)	check that o1 has been put on or. in, *according to r*, o2. (V)
CHECKSTACKUP(o)	initiate the CHECKSTACKUP2 check. (V, M)
CHECKSTACKUP2(o)	check that o has been stacked up according to the STACKUP command. (V)
CHOICECOUNT(n)	*the most recent* choicepoint is the n'th. (W, M)
CHOICETIME(n1,n2)	the n1'th choicepoint is at EVENTTIME n2. (W)
CLEAROFF(g,o)	clear off the top of o. (W, Q)
CLEARTOP(o)	o has a clear top, with no other objects on it. (Q, W)
CONJBOUND(w)	a noun-phrase boundary at a conjunction (AND) has been reached in sentence w. (B, G)
CONVIND(r,o)	compute and convert computable relations r of o to explicit HASINDRELs. (F, B)
ERSFINDNEARPAIR(n,o)	erase all FINDNEARPAIR instances with *corresponding* n and o arguments. (Q)
ERSFINDPOSS(t)	erase the FINDPOSS instances for t. (B)
ERSGETRIDCHOICES(n,g)	erase the corresponding GETRIDCHOICE instances. (W)
ERSPACKCHOICES(n,g)	erase the *corresponding* PACKCHOICE instances. (W)
ERSPUTON1CHOICES(n,g)	erase the *corresponding* PUTON1CHOICE instances. (W)
ERSREMDHASREL(o1,r,o2,s)	erase the corresponding REMDHASREL. (Q)
ERSTRIEDPACK(o,c)	erase the corresponding TRIEDPACK instances. (W)
ERSTRIEDPUT(o,c)	erase the corresponding TRIEDPUT. (W)
ERSTRIEDSTACK(o,c)	erase the corresponding TRIEDSTACK. (W)
ERSUNEVENT(n1,n2)	erase UNEVENT for n1 while backing up (BACKUP) to choicepoint n2. (W)
EVENTTIME(n)	the current event is the n'th (all events take one unit of "time"). (Q, W, M)
EXPECTMOD(w1,w2)	sentence w1 has the expectation that a modifier (UP, DOWN, etc.) w2 will occur. (T, F, M, G)
FAIL(g)	g has failed. (W)
FAILLOCATE(o)	space could not be *located for o* (which is o2 in FINDSPACE). (Q, W)
FAILPACKUP(g,o1,o2,c)	the *second major step* of PACK failed, namely trying to put o1 on an object just "packed" onto o2; goal g is the main PACK goal, of set c onto o2. (W)
FAILPUTON1(g,o1,o2)	the goal to PUTON1 o1 onto o2 fails. (W)
FAILPUTONSET(g,c,o)	the goal g to put one of the objects in set c on o has failed. (W)

● Letters in parentheses after a definition are initials of P groups in which the predicate is used.

FAILPUTONSETALL(g,c,o)	the goal g to put whole set c (as opposed to an element of it, see FAILPUTONSET) on o has failed. (W)
FAILPUTONSTACK(g,o1,o2,c)	the goal g to put o1 onto o2 in building stack c failed. (W)
FINDHIGHX(o,x1,x2,y1,y2,z)	find objects in region (x1, y1) to (x2, y2), at height z, ignoring o; objects are desired such that they bound, or close in, the region from above, with respect to the X dimension. (Q)
FINDHIGHY(o,x1,x2,y1,y2,z)	find objects as for FINDHIGHX, but in the Y dimension. (Q)
FINDLOWPAIR(n,o,x1,y1,x2,y2,z,x3,y3,sx,sy,sz)	find the lower corner of an open space in the horizontal region (x1, y1) to (x2, y2) at height z, starting from the randomly chosen point (x3, y3); ignore the space occupied by o; n is a counter which blocks this action if negative. (Q)
FINDLOWX(o,x1,x2,y1,y2,z)	find objects in region (x1, y1) to (x2, y2), at height z, ignoring o; objects are desired such that they bound, or close in, the region from below, with respect to the X dimension. (Q)
FINDLOWY(o,x1,x2,y1,y2,z)	find objects as for FINDLOWX, except in the Y dimension. (Q)
FINDNEARPAIR(n,o,x,y)	(x, y) is a candidate point for the closest object-boundary point to a point (in FINDLOWPAIR, (x3, y3)) that was randomly selected and found to be within some object; all such are examined to determine the closest, for use in a new FINDLOWPAIR attempt. (Q)
FINDSPACE(o1,o2,sx,sy,sz)	find a region of clear space on o1, ignoring space occupied by o2, of size (sx, sy, sz). (Q, W)
FOUNDHIGHPAIR(n,o,x,y,z)	collect the results of the GROWTOFIT process; n and o as in GROWTOFIT; (x, y, z) is the lower corner point of the region being examined. (Q)
FOUNDHIGHPAIR0(n,o,x,y,z)	initiate the FOUNDHIGHPAIR process. (Q)
FOUNDSPACE(o1,o2,x,y,z)	the region with lower left-hand corner at (x, y, z) is the result of FINDSPACE on o1 for o2. (Q, W)
GETRIDCHOICE(n1,g,n2,o1,o2,x,y,z)	in doing GETRIDOF o2 by putting it on o1, the point (x, y, z) has been a choice, within choicepoint number n1; this is the n2'th choice at this choicepoint. (W)
GETRIDOF(g,o)	find a place to put o other than where it is. (W, Q)
GETRIDPUT(g,o1,o2)	in trying to GETRIDOF o1, the second step is to put it on o2. (W)
GRASP(g,o)	grasp o with the hand. (Q, W)
GRASP1(g,o,x,y,z)	perform the actual movement to get the hand in position to grasp o. (Q)
GRASP2(g,o)	complete the grasp operation with GRASPING. (Q)
GRASP3(h,o)	for purposes of backing up, do an abbreviated (without the checks and subgoals) version of GRASP. (Q)
GRASPING(h,o)	h is grasping o. (Q, T, M, V)
GROWTOFIT(n,o,x1,y1,x2,y2,z,x3,y3,sx,sy,sz)	the second step of the LOCATESPACE process, the first being FINDLOWPAIR; arguments as for FINDLOWPAIR; this step tests whether there is enough space to fit the desired clear space without obstruction at the point found by FINDLOWPAIR; if so, it tries to determine a bigger region containing the sufficient clear space; see FINDHIGHX. (Q)
GROWTOFIT0(n,o,x1,y1,x2,y2,z,x3,y3,sx,sy,sz)	initiate the GROWTOFIT process; arguments as for GROWTOFIT. (Q)
GSI(w)	sentence w is an imperative. (G, F, B, M)
HASINDREL(o1,r,o2)	o1 has an indirect relation r to o2. (W, F, B)
HASLEVEL(g,n)	g has indentation level (depth) n. (Q, W, M)
HASREL(o1,r,o2,s)	o1 has relation r to o2; sign s is assumed POS for WBlox. (Q, W, E, F, B, V, M)
HASSIZE(o,sx,sy,sz)	o has size along the three co-ordinates (sx,sy,sz). (Q, W)
HASSUPERGOAL(g1,g2)	g1 has supergoal g2. (W)
HIGHX(n,o,x)	the upper X coordinate as desired by FINDHIGHX (o and n as in FINDLOWPAIR) is x. (Q)
HIGHY(n,o,y)	similar to HIGHX, except for the Y dimension. (Q)
IMPCHOICE(o)	o has been used as a choice for an indefinite determiner in an imperative sentence. (B)
IMPCHOOSE(t)	choose a referent for t, in an imperative sentence. (B)
IMPINDEF(p)	the word at p is an indefinite determiner, in an imperative sentence. (N, B, G)
IMPOBJ(w,o)	the main object (or one of a set) for the imperative sentence w is o. (M, B)
IMPREL(w,r,o)	the relation to be fulfilled in imperative sentence w is r of o1. (M, T, G)

IMPRESTR(t,o1,r,o2)	a possible alternative for t as the main object in an imperative sentence is o1, in conjunction with relation r of o2. (M, F)
IMPTYPE(w1,w2)	the type of imperative in sentence w1 is w2 (PICKUP, etc.). (M, G)
INSET(o,c)	o is in set c. (W)
INSTACK(o,c)	o is in c; stacks are loosely defined to include trees of blocks. (Q, V)
ISCOMPREL(r)	r is a computable relation. (F, B, T)
ISIMPER(p)	the word at p is an imperative grammar type. (N, T, G)
ISINDREL(r)	r is an indirect relation. (F, B, T)
LOCAT(o,x,y,z)	o has its lower left-hand corner at (x,y,z); o may also be the hand. (Q, F, M, V)
LOCATERESULT(o,x1,y1,x2,y2,z)	the region found by the LOCATESPACE process is (x1, y1, z) to (x2, y2, z); o is the object ignored in that process. (Q)
LOCATESPACE(o1,o2,sx,sy,sz)	initiate the actual process of finding space; see FINDSPACE. (Q, W)
LOWX(n,o,x)	the lower X coordinate as desired by FINDLOWX (o and n as in FINDLOWPAIR) is x. (Q)
LOWY(n,o,y)	similar to LOWX, except for the Y dimension. (Q)
MAKESPACE(g,o1,o2,sx,sy,sz)	make space on o1 ignoring space occupied by o2, of size (sx, sy, sz). (Q, W)
MAKESPACE2(g,o1,o2,sx,sy,sz)	the second step in the MAKESPACE process, to try to find space after removing an object from o1; arguments as for MAKESPACE. (Q)
MAKESPACE3(g,o1,o2,sx,sy,sz)	the final step in MAKESPACE, which detects success or repeats the whole process; arguments as for MAKESPACE. (Q)
MOVEHAND(x,y,z)	move hand to (x,y,z). (Q)
NEWLOCAT(o)	o is at a new location; remove any old relations that are no longer valid. (Q)
NEWLOCAT2(o)	o is at a new location; add any new relations that hold. (Q)
NEXT(g,w)	when g succeeds, assert w. (W, Q)
NEXTF(g,w)	when g fails, assert w. (W)
NOCLEAR(g)	the present PUTON process involves a set, so inhibit clearing away objects that seem to be in the way. (Q, W)
NPGCHK1(p)	check for noun-phrase grammar adjacencies at p; first step is actual checks. (N)
NPGCHK2(p)	a delayed initiation of the second step in checking noun-phrase grammar. (N)
NPGCHK3(p)	perform the second step of the noun-phrase grammar check at p, which is to signal error if appropriate. (N)
NREPLY(n)	the number of replies so far is n. (V, S)
PACK(g,c,o)	pack the objects in set c onto o. (W)
PACKCHOICE(n1,g,n2,o1,o2,x,y,z)	the n2'th choice at choicepoint n1, trying g, is to PACK o1 on o2 at (x, y, z). (W)
PACKPUT(g,c,o1,o2)	the PUT step of PACK goal g of set c onto o2 is to place object o1. (W)
PACKUPON(g,c,o1,o2)	the second major step of g, PACKing c onto o2, is to try to put something from c onto o1. (W)
PICKUP(g,o)	pick up o. (W, M)
PICKUP2(g,h)	the finishing step in the PICKUP process is to be done, i.e., raising h. (W)
PUT(g,o,x,y,z)	put o at (x,y,z). (Q, W)
PUTDOWN(g,o)	put o down on the table or wherever there is space. (W, M)
PUTMOVE(g,o,x,y,z)	do the actual movement of the hand associated with a PUT. (Q)
PUTON(g,o1,o2)	put o1 on o2; there may be a set of instances with the same o2 argument (see PUTONSET). (W, M)
PUTON1(g,o1,o2)	put the single object o1 on o2, as opposed to PUTON, which might become PUTONSET. (W)
PUTON1CHOICE(n1,g,n2,o1,o2,x,y,z)	the n2'th choice at g, a PUTON1 goal of o1 onto o2, choicepoint n1, is the location (x, y, z). (W)
PUTONPUT(g,o1,o2)	start the actual PUT step of a PUTON1 goal. (W)
PUTONSET(g,c,o)	put objects in set c on o by choosing and using PUTON1 iteratively. (W)
PUTONSETO(c)	collect the set of objects in instances of PUTON for PUTONSET. (W)
PUTONSETCHOICE(n,g,c,o)	the choicepoint n for PUTONSET involves putting set c on o. (W)
RAISEHAND(h)	raise the hand by moving it straight up. (Q, W)
RELRESTR1(o1,p,r,o2,s)	perform the first step in the relation-restriction process, which is to check for possible need for IMPRESTR, q.v. (F)
RELRESTR2(t,p,r,o,s)	the second step in the relation-restriction process, which is to check r of o for possible referents for t, to restrict the set. (F)

RELRESTRCHK(t,p,r,o,s)	the former RELRESTRCHK is now RELRESTRCHK2; this now signals a preliminary step to the check-relation-restriction process, which first checks whether the relation at hand is an indirect or computable one. (B)
RELRESTRCHK2(t,p,r,o,s)	check whether the corresponding RELRESTR should be applied. (B)
REMDHASREL(o1,r,o2,s)	the relation (o1,r,o2) has been removed; update INSTACK relations affected. (Q)
REMDINSTACK(o,c)	o has been removed from c; check if anything remains of c except the bottom block. (Q)
REPLV(n,w)	the n'th reply (in order of generation) is w. (V)
REPLVO(w)	w is a new reply, yet to be counted (see REPLV). (V, E, M, D)
RETRY(g)	g is to be retried, i.e., restarted after a BACKUP, with a new choice made. (W)
STACKSET(c)	collect the objects from STACKUP instances into set c. (W)
STACKUP(g,o)	stack up o as part of a set of such instances. (W, M)
STACKUPSET(g,c)	stack up the objects in set c. (W)
SUCCEED(g)	g has succeeded; continue appropriately. (W, Q)
TRACEPUTIN(w)	print a trace message for the PUTIN command; w is a dummy. (M)
TRIEDPACK(o,c)	in PACKing the set c onto somewhere, object o has now been tried. (W)
TRIEDPUT(o,c)	in putting c on some object, o has been tried. (W)
TRIEDSTACK(o,c)	in stacking up objects in set c, o has been tried. (W)
UNEVENT(n,w)	the way to undo the event at EVENTTIME n is w. (W, Q)
UNGRASP(o)	let go of o, from the hand. (Q)
USERESULT(o1,o2,sx,sy,w)	use the open region found by LOCATESPACE process, which should be of size (sx, sy) on the horizontal plane, in the way specified by w, which is one of {PACK, RANDOM, CENTER}. (Q, W)
WBPINIT(g)	initialize for starting up the WBlox Ps, top level goal g. (M)

F. Summary and Discussion

MiliPS represents a successful implementation in PSs of a language system with some general features, namely, objects with relations and attributes, main sentence forms that describe a scene, imperative forms, and a variety of queries. Inputs are processed without recourse to conventional syntactic parsing, and no tree-structured representation of them is formed. Text is converted immediately on being scanned to an internal form, which is quite sufficient for further manipulations, but which doesn't preserve the surface structure at all. At each point in the left-to-right scan of an input, as much as possible is known and inferred from what has been scanned. Five forms of completeness have been discussed, and MiliPS's capabilities were delineated with respect to those, providing a measure of its potential performance beyond the 50 test sentences exhibited. Linguistic anomalies are systematized into the categories ambiguity, redundancy, and inconsistency, and the main reaction of the system to inputs is based on the interaction of sentence type and the presence of those anomalies. Augmenting an early version to handle the blocks manipulation task was carried out by major additions to the set of Ps with few changes to existing Ps and with no deletions.

WBlox is a specialized problem-solver for blocks manipulations of a simple sort. Its organization is hierarchical, it features operating on a model and carrying out updating procedures as a result of operations, and is capable of backtracking in a search space to find a feasible plan of action. The system's goals are explicit and are sequenced to result in search behavior representable as an and-or graph. A less prominent backtracking mechanism is needed here than in the original Planner implementation of a similar blocks problem solver. Analysis of the problem domain allowed some decisions made formerly by backtracking to become more precise ordering decisions, taking advantage of selectivity in the LHSs of Ps. The remaining decisions requiring potential backtracking were formulated explicitly as choicepoints and associated with a stream of undoable primitive operations, rather than having mechanisms of questionable flexibility built into the underlying architecture as in Planner and other recent AI languages. A set of tests were devised to fully exercise the capabilities of the problem-solving system.

The question of whether the present system, and more generally PSs, could be used for further research can be approached along the lines of the language system and the problem-solving system independently. The completeness considerations in Section B.1 support a wide task domain coverage for the present system and indicate a framework for making additions to the system to rationally order the priorities for augmentation. The precise formulation of semantic cases, discussed in Section B.2, Section B.3, and Section D.4 raise further issues for augmentation and indicate how minor some of the omitted considerations in the present system are. Nevertheless, analysis of the existing cases, explicitly given as Ps and thus in a usable form, might be fruitful for cleaning up the structure and giving it more inherent generality and flexibility.

To use the present techniques in a new task domain would first require a new lexicon, which simply involves changing the tagging process (T Ps), which are independently modifiable. It would probably be necessary to augment the grammatical adjacency tests for new word classes, but this doesn't present obvious difficulties either. The semantics of blocks relations and how relations interact in the understanding of inputs might be the area requiring the most new problem-solving. There is already present a system of dividing relations into direct ones, indirect ones, and computable ones, and that scheme and its processing conventions might carry over intact (cf. the discussion in Section D.2). The actual use of relations in referent determination would probably be along the lines of the present F Ps, but considerations there would probably not interact with the closely related set of B Ps, given that many interactions have already been worked out in response to the demands of the augmentation included in the present work.

Some further work has already been done by others within the basic blocks problem solving domain. In particular, Fahlman (1974) describes a reworking and extension of the blocks task, which in retrospect might have served as a better vehicle for comparisons than the original one used here. Of the nature of the blocks tasks that he focussed on, it suffices to say that they involved building more complex structures than in WBlox, sometimes using auxiliary structures, allowing rotations of objects, enabling intercommunication between goals, and modelling the mechanics of contact and balance of objects more carefully. Fahlman developed a flexible control structure within the Conniver framework (Sussman and McDermott, 1972), and asserted its superiority over Planner and similar languages, and also specifically over PSs. Fahlman emphasized the importance of being able to: set up explicit goals; test hypotheses; switch back and forth between alternate promising approaches to a goal; and give up on an approach with specific difficulties communicated back to higher goals. Of those four features, only the last is something that hasn't yet been explicitly demonstrated in PSs, although keeping major alternative approaches for relatively large models also deserves further research in the PS framework. I will now discuss some of Fahlman's points in more detail, and argue that the ability of PSs to grapple with the difficulties of the task domain is promising, if not already demonstrated.

Fahlman developed a "choice-gripe" control structure, in which each choicepoint is explicit and sets up a gripe handler so that failures of subgoals following the choice, when those failures include specific gripes on why they occurred, can be processed appropriately. A gripe handler reacts more flexibly than choicepoint recovery in WBlox, in that it can involve taking better preparatory steps and then retrying the subgoal, or redefining the subgoal in some way and then retrying it, or taking other similar corrective actions. It seems clear that the present choicepoints in WBlox could easily be extended to behave in these more flexible ways, according to task demands, since the recovery is handled by specific Ps.

In trying alternative paths, Fahlman made use of Conniver's multiple data contexts, in which context tags are used to point to complete context alternatives, allowing them to be examined, resumed, or suspended. Such a facility, if the task really required it (as opposed to using it as a convenience because it's there), would be an explicit mechanism in PSs, perhaps storing alternative contexts as Ps and having them selectively evokable for examination or resumption. But a PS approach might be found to avoid that by coding, instead, methods for patching up difficulties or revising an ever-current state to make it look as if something different had been done. Based on the limited evidence on human behavior, e.g. in Newell and Simon (1972), humans seem to make use of mistakes without having to return completely to a state on some other branch of a search tree, and perhaps have better diagnostic and recovery methods because of limitations along the same lines as would be the case in a PS implementation. Rather than storing entire states, the alternative might be to keep path information so that a previously-seen knowledge state could be recomputed (perhaps laboriously) if necessary.

Several minor topics raised by Fahlman can now be discussed. His system made use of a distinction between primary data and secondary data, which can be re-computed if necessary from the primary data, but which is kept around anyway, subject to erasure if storage becomes scarce. This might correspond to having a fading Working Memory, where items not accessed for some period of time simply disappear. Such a scheme has not been implemented, but it has been indicated as useful in several places in the present work. Fahlman additionally proposes that memory fade be based on the difficulty of recomputation and on some estimate of expected usefulness. Fahlman comments on the overall loose style of his system, which allows it to step back from local jam-ups and try to get around them. This is just as much an attribute of PSs, given the appropriate memory representation of what constitutes a jam-up. He says his program is prone to get into infinite loops, and proposes that a more sophisticated system would record states and occasionally check back to make sure there isn't serious repetition. Such a solution should be equally feasible in PSs, although perhaps not as necessary because PS architectures have some built-in safeguards, e.g. not firing a P on the same data twice unless some of it has been re-asserted. The topic of loops needs further research, certainly. Finally, the goal intercommunication in Fahlman's system, which includes both protection of goals' results from interference and dissemination of useful information to others, should be quite feasible in PSs due to the open, global nature of the Working Memory.

G. References

Card, S., Rubin, A. and Winograd, T., 1972. "Provisional SHRDLU users' manual", Pittsburgh, Pa: Carnegie-Mellon University, Department of Computer Science. Version 0.

Fahlman, S. E., 1974. "A planning system for robot construction tasks", *Artificial Intelligence*, Vol. 5, 1, pp. 1-49.

Hays, D. G., 1964. "Dependency theory: A formalism and some observations", *Language*, 40, 4. pp. 511-525.

Hewitt, C., 1969. "Planner: a language for proving theorems in robots", in Walker, D. E. and Norton, L. M., Eds., *Proc. First International Joint Conference on Artificial Intelligence*, pp. 167-301. Boston, Ma: The Mitre Corp..

Hewitt, C., 1971. "Procedural embedding of knowledge in Planner", *Proc. Second International Joint Conference on Artificial Intelligence*, pp. 167-182.

Hewitt, C., 1972. "Description and theoretical analysis (using schemata) of Planner: A language for proving theorems and manipulating models in robots", TR-258. Cambridge, Ma: MIT Artificial Intelligence Laboratory.

Moran, T. P., 1972. "MILISY: the mini-linguistic system", in Newell, A., Reddy, R., et. al., Eds., *CSD Artificial Intelligence Study Guide 72*, pp. 3.23-3.45. Pittsburgh, Pa: Carnegie-Mellon University, Department of Computer Science.

Newell, A. and Simon, H. A., 1972. *Human Problem Solving*, Englewood Cliffs, NJ: Prentice-Hall.

Pratt, V. R., 1975. "LINGOL - A progress report", *Proc. Fourth International Joint Conference on Artificial Intelligence*, pp. 422-428.

Sussman, G. J., 1973. "A computational model of skill acquisition", AI TR-297. Cambridge, Ma: MIT Artificial Intelligence Laboratory.

Sussman, G. J. and McDermott, D. V., 1972. "Why Conniving is better that Planning", Memo 255A. Cambridge, MA: MIT Artificial Intelligence Laboratory.

Sussman, G. J. and Winograd, T., 1970. "Micro-planner reference manual", Memo 203. Cambridge, MA: MIT Artificial Intelligence Laboratory.

Winograd, T., 1972. *Understanding Natural Language*, New York, NY: Academic Press.

MILIPS/WBLOX APPENDICES

Appendix A. MILIPS PROGRAM LISTING

```
BEGIN          % PS FOR MILISY %

EXPR MILIPS(): BEGIN NONFLUENT(LEFTOF); REQUIRE(MILGAMP,MILN,MILF,MILM,MILVD);
     DCMD(MILC);

     % P GROUPS: S, T, E, G, A, R, P, N, F, B, M, V, D, X %

   % MACROS:
     LEXORDER(A,B) TESTS IF A IS LEXICALLY LESS THAN OR EQ B
     SAY: CONVERTS AN UNEVALUATED ARGUMENT AS SAYQ DOES: THE ARGUMENT
          IS TAKEN AS THE CONS OF SAYQ'S TWO ARGUMENTS
     SAYQ(13,'(A BIG CLOCK DANCES)) --
      EXISTS(S,LE,RE,A1,B2,C3,D4) & SCANFIN(LE) & SENTENCE(S)
      & ENDMARK(LE) & ENDMARK(RE) & TEXT(13,'(A BIG CLOCK DANCES))
      & LEFTOF(LE,A1) & EQA(A1) & LEFTOF(A1,B2) & EQBIG(B2) & LEFTOF(B2,C3)
      & EQCLOCK(C3) & LEFTOF(C3,D4) & EQDANCES(D4) & LEFTOF(D4,RE);
     TRACEPRINTM PRINTS ITS ARGUMENT AS A MESSAGE
   %
     % S = SCANNING, T = TAGGING, E = ERROR AND EXTERNAL TRACE %

S0: "SCAN LE" = SCANFIN(X) & ENDMARK(X) & LEFTOF(X,Y) & TEXT(N,Z)
   -> SCAN(Y) & SCANFIN(Y) & NEGATE(1)
     & TRACING(TRACEPRINTM(N CONS '(INPUT TEXT IS ?") @ Z @ '(?") ));
S1: "SCAN ON" = SCANFIN(X) & LEFTOF(X,Y) & NOT ENDMARK(X) & NOT ENDMARK(Y)
     & NOT SCAN(X)
   -> SCAN(Y) & SCANFIN(Y) & NEGATE(1);
S4: "SCAN FIN" = SCANFIN(X) & LEFTOF(X,Y) & ENDMARK(Y) & NOT SCAN(X)
     & SENTENCE(S)
   -> NPBOUND(Y) & SENTBOUND(S) & NEGATE(1);
S7: "SCAN ERR" = SCANFIN(X) & NOT( NOT SCAN(X) & NOT SCAN(X) ) & LEFTOF(Y,X)
   -> ERROR(Y,'(LEXICAL)) & NEGATE(1) & NOT SCAN(X);

T1: "TAG COP" = SCAN(X) & EQIS(X) & LEFTOF(X,Y) & NOT EQNOT(Y)
   -> ISCOP(X,'POS) & WORDEQ(X,'IS) & NEGATE(1,2);
T2: "SKIP COP" = SCAN(X) & EQIS(X) & LEFTOF(X,Y) & EQNOT(Y) -> NEGATE(1);
T4: "TAG COP NEG" = SCAN(X) & EQNOT(X) & LEFTOF(W,X) & EQIS(W) & LEFTOF(V,W)
   -> ISCOP(X,'NEG) & LEFTOF(V,X) & WORDEQ(X,'ISNOT) & NEGATE(ALL);

T7: "TAG COLOR1" = SCAN(X) & EQRED(X)
   -> ISAVW(X,'COLOR,'RED) & WORDEQ(X,'RED) & NEGATE(ALL);
T10: "TAG COLOR2" = SCAN(X) & EQGREEN(X)
   -> ISAVW(X,'COLOR,'GREEN) & WORDEQ(X,'GREEN) & NEGATE(ALL);
T13: "TAG COLOR3" = SCAN(X) & EQBLUE(X)
   -> ISAVW(X,'COLOR,'BLUE) & WORDEQ(X,'BLUE) & NEGATE(ALL);
T16: "TAG COLOR4" = SCAN(X) & EQBLACK(X)
   -> ISAVW(X,'COLOR,'BLACK) & WORDEQ(X,'BLACK) & NEGATE(ALL);

T21: "TAG SIZE1" = SCAN(X) & EQLARGE(X)
   -> ISAVW(X,'SIZE,'LARGE) & WORDEQ(X,'LARGE) & NEGATE(ALL);
T24: "TAG SIZE2" = SCAN(X) & EQMEDIUM(X)
   -> ISAVW(X,'SIZE,'MEDIUM) & WORDEQ(X,'MEDIUM) & NEGATE(ALL);
T27: "TAG SIZE3" = SCAN(X) & EQSMALL(X)
   -> ISAVW(X,'SIZE,'SMALL) & WORDEQ(X,'SMALL) & NEGATE(ALL);

T31: "TAG REL1" = SCAN(X) & EQIN(X)
   -> ISRELW(X,'IN) & WORDEQ(X,'IN) & NEGATE(ALL);
T34: "TAG REL2" = SCAN(X) & EQON(X)
   -> ISRELW(X,'ON) & WORDEQ(X,'ON) & NEGATE(ALL);
T37: "TAG REL3" = SCAN(X) & EQNEAR(X)
   -> ISRELW(X,'NEAR) & WORDEQ(X,'NEAR) & NEGATE(ALL);
T39: "TAG REL4" = SCAN(X) & EQUNDER(X)
   -> ISRELW(X,'UNDER) & WORDEQ(X,'UNDER) & NEGATE(ALL);

T41: "TAG NOUN1" = SCAN(X) & EQBALL(X)
   -> ISNOUNW(X,'BALL) & WORDEQ(X,'BALL) & NEGATE(ALL);
T44: "TAG NOUN2" = SCAN(X) & EQBLOCK(X)
   -> ISNOUNW(X,'BLOCK) & WORDEQ(X,'BLOCK) & NEGATE(ALL);
T47: "TAG NOUN3" = SCAN(X) & EQTABLE(X)
   -> ISNOUNW(X,'TABLE) & WORDEQ(X,'TABLE) & NEGATE(ALL);
T50: "TAG NOUN4" = SCAN(X) & EQFLOOR(X)
   -> ISNOUNW(X,'FLOOR) & WORDEQ(X,'FLOOR) & NEGATE(ALL);
T53: "TAG NOUN5" = SCAN(X) & EQBOX(X)
   -> ISNOUNW(X,'BOX) & WORDEQ(X,'BOX) & NEGATE(ALL);

T57: "TAG NOUNQ" = SCAN(X) & EQWHAT(X) & LEFTOF(W,X) & ENDMARK(W)
   -> QNOUN(X) & ISNOUNW(X,'WHAT) & WORDEQ(X,'WHAT) & NEGATE(1,2);
T60: "REL PRON" = SCAN(X) & EQWHICH(X)
   -> ISRELPRONW(X) & WORDEQ(X,'WHICH) & NEGATE(ALL);
T63: "REL PRON" = SCAN(X) & EQTHAT(X)
   -> ISRELPRONW(X) & WORDEQ(X,'THAT) & NEGATE(ALL);
```

```
E2: ERROR(X,R) -> ERRORS(X,'(???) & REPLY(R) & NEGATE(1);
E4: ERRORS(X,EL) & LEFTOF(Y,X) & NOT ENDMARK(Y) & WORDEQ(X,XW)
   -> ERRORS(Y,XW CONS EL) & NEGATE(1);
E6: ERRORS(X,EL) & LEFTOF(Y,X) & ENDMARK(Y) & WORDEQ(X,XW)
   -> REPLY(XW CONS EL) & NEGATE(1);
E8: ERRORS(X,EL) & ERRREF(X,L) -> ERRORS(R,EL) & NEGATE(ALL);

E11: "TRACE AV" = HASAV(O,A,V,I)
   -> TRACING(TRACEPRINTM(<'ADDING,A,V,<I>,'TO,O>));
E12: "TRACE REL" = HASREL(O,R,O2,S)
   -> TRACING(TRACEPRINTM(<'ADDING,O,R,O2,<S>>));
E13: "TRACE ISA" = ISA(O,N) -> TRACING(TRACEPRINTM(<'ADDING,N,O>));

E21: "TRACE P INC" = PREDINCONT(O,X,A,V,S)
   -> PREDINCON(O,X,A,V,S) & TRACING(TRACEPRINTM(<'PREDINCON,O,X,A,V,S>));
E22: "TRACE P RED" = PREDREDUNT(O,X,A,V,S)
   -> PREDREDUN(O,X,A,V,S) & TRACING(TRACEPRINTM(<'PREDREDUN,O,X,A,V,S>));
E23: "TRACE P RESTR" = PREDRESTRT(O,X,A,V,S)
   -> PREDRESTR(O,X,A,V,S) & TRACING(TRACEPRINTM(<'PREDRESTR,O,X,A,V,S>));

E31: "TRACE R INC" = RELINCONT(O,X,A,V,S)
   -> RELINCON(O,X,A,V,S) & TRACING(TRACEPRINTM(<'RELINCON,O,X,A,V,S>));
E32: "TRACE R RED" = RELREDUNT(O,X,A,V,S)
   -> RELREDUN(O,X,A,V,S) & TRACING(TRACEPRINTM(<'RELREDUN,O,X,A,V,S>));
E33: "TRACE R RESTR" = RELRESTRT(O,X,A,V,S)
   -> RELRESTR(O,X,A,V,S) & TRACING(TRACEPRINTM(<'RELRESTR,O,X,A,V,S>));

END;

                    ......................

     % G = TOP-LEVEL GRAMMAR, A = ADJECTIVES %          % PAGE 2 %

EXPR MILGAMP(): BEGIN

G1: "THE" = SCAN(X) & EQTHE(X) & SENTENCE(S) & GTYPED(S)
   -> DEFDET(X) & WORDEQ(X,'THE) & NEGATE(1,2);
G2: "THE INIT" = SCAN(X) & EQTHE(X) & SENTENCE(S) & NOT GTYPED(S)
   -> DEFDET(X) & GSD(S) & GTYPED(S) & WORDEQ(X,'THE) & NEGATE(1,2);
G5: "A DEF" = SCAN(X) & EQA(X) & SENTENCE(S) & GSQE(S) & LEFTOF(W,X)
     & WORDEQ(W,WW) & SATISFIES(WW,WW EQ 'THERE)
   -> DEFDET(X) & WORDEQ(X,'A) & NEGATE(1,2);
G6: "A IND" = SCAN(X) & EQA(X) & SENTENCE(S) & GTYPED(S) & NOT GSQE(S)
   -> INDEFDET(X) & WORDEQ(X,'A) & NEGATE(1,2);
G7: "A INIT" = SCAN(X) & EQA(X) & SENTENCE(S) & NOT GTYPED(S)
   -> INDEFDET(X) & GTYPED(S) & GSD(S) & WORDEQ(X,'A) & NEGATE(1,2);
G9: "THERE" = SCAN(X) & EQTHERE(X) & SENTENCE(S) & NOT GTYPED(S)
   -> GSE(S) & GTYPED(S) & WORDEQ(X,'THERE) & NEGATE(1,2);

G10: "THERE Q" = SCAN(X) & EQTHERE(X) & GSQE(S) & LEFTOF(W,X) & ISCOP(W,I)
   -> WORDEQ(X,'THERE) & NEGATE(1,2);
G13: "WHAT Q" = QNOUN(X) & ISNOUNW(X,XW) & SATISFIES(XW,XW EQ 'WHAT)
     & SENTENCE(S) & NOT GTYPED(S)
   -> GSQW(S) & GTYPED(S) & EXISTS(OBJ) & QWFIND(OBJ,X) & CUROBJ(OBJ,'MAIN)
     & CUROBJP(OBJ,'MAIN) & ISNOUN(X,XW) & ERRREF(OBJ,X) & NEGATE(1,2);
G17: "IS Q" = ISCOP(X,I) & SENTENCE(S) & NOT GTYPED(S) & LEFTOF(X,Y)
     & EQTHERE(Y)
   -> GSQE(S) & GTYPED(S);
G18: "IS Q" = ISCOP(X,I) & SENTENCE(S) & NOT GTYPED(S) & LEFTOF(X,Y)
     & NOT EQTHERE(Y)
   -> GSQO(S) & GTYPED(S);
G21: "WHERE" = SCAN(X) & EQWHERE(X) & SENTENCE(S) & NOT GTYPED(S)
   -> GSQWR(S) & GTYPED(S) & WORDEQ(X,'WHERE) & NEGATE(1,2);

G31: "COP -" = ISCOP(X,I) & SATISFIES(I,I EQ 'NEG)
   -> COPSIGN('NEG) & NOT COPSIGN('POS);
G32: "COP +" = ISCOP(X,I) & SATISFIES(I,I EQ 'POS)
   -> COPSIGN('POS) & NOT COPSIGN('NEG);

A1: "AV WFND" = ISAV(X,A,V,I) & NOT OLDAV(X) & CUROBJ(O,P) & ISDEF(S)
   -> AVRESTR(O,X,A,V,I) & OLDAV(X);
A5: "AV NEW" = ISAV(X,A,V,I) & NOT OLDAV(X) & CUROBJ(O,P) & ISINDEF(S)
   -> NEWAV(O,A,V,I) & OLDAV(X);
A14: "AV G1" = ISAVW(X,A,V) & LEFTOF(W,X) & ISNOUN(W,WW) & GSQO(S)
     & LEFTOF(X,Y) & ENDMARK(Y)
   -> ANSPREDFIN(A,V,'POS) & NEGATE(1);
A15: "AV G2" = ISAVW(X,A,V) & LEFTOF(W,X) & ISAV(W,A2,V2,I2) & NOT ISPRED(W)
   -> ISAV(X,A,V,'POS) & NEGATE(1);
A17: "AV G4" = ISAVW(X,A,V) & LEFTOF(W,X) & ISCOP(W,I)
   -> ISPRED(X) & ISAV(X,A,V,I) & NEGATE(1);
A18: "AV G6" = ISAVW(X,A,V) & LEFTOF(W,X) & DETREEN(W)
```

```
    -> ISAV(X,A,V,'POS) & NEGATE(1);
A28: "AV GFAIL" = ISAVW(X,A,V) & LEFTOF(W,X) & NOT( EXISTS(I) & ISCOP(W,I) )
       & NOT( EXISTS(A2,V2,I2) & ISAV(W,A2,V2,I2) & NOT ISPRED(W) )
       & NOT DETSEEN(W)
       & NOT( EXISTS(Y,S,WW) & GSQD(S) & ISNOUN(W,WW) & LEFTOF(X,Y)
              & ENDMARK(Y) )
    -> ERROR(X,'(GRAMMAR));


       % R = RELATIONS (PREPOSITIONS), P = RELATIVE PRONOUNS %

R1: "REL G1" = ISRELW(X,XW) & LEFTOF(W,X) & ISCOP(W,I)
    -> ISREL(X,XW) & NEGATE(1);
R2: "REL G2" = ISRELW(X,XW) & LEFTOF(W,X) & ISNOUN(W,WW)
    -> ISREL(X,XW) & NEGATE(1);
R3: "REL G3" = ISRELW(X,XW) & LEFTOF(W,X) & ISPRED(W)
    -> ISREL(X,XW) & NEGATE(1);
R5: "REL GFAIL" = ISRELW(X,XW) & LEFTOF(W,X) & NOT( EXISTS(WW) & ISNOUN(W,WW) )
       & NOT( EXISTS(I) & ISCOP(W,I) ) & NOT ISPRED(W)
    -> ERROR(X,'(GRAMMAR));
R11: "REL NOTE" = ISREL(R,RW) & NOT OLDREL(R) & CUROBJ(O,P) & COPSIGN(I)
    -> HASRELN(O,RW,I) & OLDREL(R) & NEGATE(4);
R12: "REL NOTE2" = ISREL(R,RW) & NOT OLDREL(R) & CUROBJ(O,P)
       & NOT( EXISTS(I) & COPSIGN(I))
    -> HASRELN(O,RW,'POS) & OLDREL(R);

P1: "RELPRON G" = ISRELPRONW(X) & LEFTOF(W,X) & ISNOUN(W,WW)
    -> ISRELPRON(X) & NEGATE(1);
P2: "RELPRON G2" = ISRELPRONW(X) & LEFTOF(W,X) & ISPRED(W)
    -> ISRELPRON(X) & NEGATE(1);
P8: "RELPRON GFAIL" = ISRELPRON(X) & LEFTOF(W,X)
       & NOT( EXISTS(WW) & ISNOUN(W,WW) ) & NOT ISPRED(W)
    -> ERROR(X,'(GRAMMAR));
END;


                       ------------------------


       % N = NOUN PHRASES AND NOUNS %                 % PAGE 3 %

EXPR MILN(); BEGIN

N1: "DEF DET" = DEFDET(X) & CUROBJ(O,OP) & NOT DETSEEN(X)
    -> NPGCHK(X) & DETSEEN(X) & EXISTS(OBJ) & DEFFND(OBJ,X) & CUROBJ(OBJ,D)
       & CUROBJP(O,OP) & ISDEF(OBJ) & NEGATE(2);
N2: "DEF DET" = DEFDET(X) & NOT( EXISTS(O,OP) & CUROBJ(O,OP) )
    -> NPGCHK(X) & DETSEEN(X) & EXISTS(OBJ) & DEFFND(OBJ,X)
       & CUROBJP(OBJ,'MAIN) & CUROBJ(OBJ,'MAIN) & ISDEF(OBJ);
N5: "INDEF DET" = INDEFDET(X) & CUROBJ(O,OP) & NOT DETSEEN(X)
    -> NPGCHK(X) & DETSEEN(X) & EXISTS(OBJ) & CUROBJ(OBJ,D)
       & CUROBJP(O,OP) & ISINDEF(OBJ) & NEGATE(2);
N6: "INDEF DET" = INDEFDET(X) & NOT( EXISTS(O,OP) & CUROBJ(O,OP) )
    -> NPGCHK(X) & DETSEEN(X) & EXISTS(OBJ) & CUROBJ(OBJ,'MAIN) & ISINDEF(OBJ);

N8A: "NP GRAM" = NPGCHK(X) & LEFTOF(W,X) & WORDEQ(W,WW)
       & SATISFIES(WW,WW EQ 'THERE) & GSQE(S) & CUROBJ(O,P) & ISDEF(O)
    -> NEGATE(1);
N8B: "NP GRAM" = NPGCHK(X) & LEFTOF(W,X) & ISREL(W,WW) -> NEGATE(1);
N8C: "NP GRAM" = NPGCHK(X) & LEFTOF(W,X) & ISCOP(W,I) -> NEGATE(1);
N8D: "NP GRAM" = NPGCHK(X) & LEFTOF(W,X) & ENDMARK(W) -> NEGATE(1);
N10: "NP UNGRAM" = NPGCHK(X) & LEFTOF(W,X) & NOT( EXISTS(WW) & ISREL(W,WW) )
       & NOT( EXISTS(S,O,P,WW) & WORDEQ(W,WW) & SATISFIES(WW,WW EQ 'THERE)
              & GSQE(S) & CUROBJ(O,P) & ISDEF(O))
       & NOT( EXISTS(I) & ISCOP(W,I) ) & NOT ENDMARK(W)
    -> ERROR(X,'(GRAMMAR)) & NEGATE(1);

N15: "NP BDC" = ISCOP(W,I) & SENTENCE(S) & GSD(S) & LEFTOF(V,W)
       & NOT ISRELPRON(V)
    -> NPBOUND(W) & NPBOUNDL(W);
N16: "NP BDC" = ISCOP(W,I) & SENTENCE(S) & GSQW(S) & LEFTOF(V,W)
       & NOT ISRELPRON(V)
    -> NPBOUND(W) & NPBOUNDL(W);

N21: "N G1" = ISNOUNW(X,XW) & LEFTOF(W,X) & ISAV(W,A,V,I)
    -> ISNOUN(X,XW) & NEGATE(1);
N22: "N G2" = ISNOUNW(X,XW) & LEFTOF(W,X) & DEFDET(W)
    -> ISNOUN(X,XW) & NEGATE(1);
N23: "N G3" = ISNOUNW(X,XW) & LEFTOF(W,X) & INDEFDET(W)
    -> ISNOUN(X,XW) & NEGATE(1);
N29: "N GFAIL" = ISNOUNW(X,XW) & LEFTOF(W,X)
       & NOT( EXISTS(A,V,I) & ISAV(W,A,V,I) ) & NOT DEFDET(W) & NOT INDEFDET(W)
    -> ERROR(X,'(GRAMMAR));
```

```
N31: "N INDEF" = ISNOUN(X,XW) & CUROBJ(O,P) & ISINDEF(O)
       & NOT( EXISTS(O2) & ERRREF(O2,X) )
    -> MAKISA(X,XW,O,P) & ERRREF(O,X) & NEGATE(2,3);
N33: "N DEF" = ISNOUN(X,XW) & CUROBJ(O,P) & ISDEF(O)
       & NOT( EXISTS(O2) & ERRREF(O2,X) )
    -> NRESTR(O,X,XW) & ERRREF(O,X);

MA1: "ISA BALL" = MAKISA(X,XW,O,P) & SATISFIES(XW,XW EQ 'BALL)
    -> EXISTS(BALL) & ADDAV(BALL,O) & ISA(BALL,'BALL) & CUROBJ(BALL,P)
       & REFERS(BALL,BALL) & ERRREF(BALL,X) & NEWOBJ(BALL) & NEGATE(1);
MA2: "ISA BLOCK" = MAKISA(X,XW,O,P) & SATISFIES(XW,XW EQ 'BLOCK)
    -> EXISTS(BLOCK) & ADDAV(BLOCK,O) & ISA(BLOCK,'BLOCK) & CUROBJ(BLOCK,P)
       & REFERS(BLOCK,BLOCK) & ERRREF(BLOCK,X) & NEWOBJ(BLOCK) & NEGATE(1);
MA3: "ISA TABLE" = MAKISA(X,XW,O,P) & SATISFIES(XW,XW EQ 'TABLE)
    -> EXISTS(TABLE) & ADDAV(TABLE,O) & ISA(TABLE,'TABLE) & CUROBJ(TABLE,P)
       & REFERS(TABLE,TABLE) & ERRREF(TABLE,X) & NEWOBJ(TABLE) & NEGATE(1);
MA4: "ISA FLOOR" = MAKISA(X,XW,O,P) & SATISFIES(XW,XW EQ 'FLOOR)
    -> EXISTS(FLOOR) & ADDAV(FLOOR,O) & ISA(FLOOR,'FLOOR) & CUROBJ(FLOOR,P)
       & REFERS(FLOOR,FLOOR) & ERRREF(FLOOR,X) & NEWOBJ(FLOOR) & NEGATE(1);
MA5: "ISA BOX" = MAKISA(X,XW,O,P) & SATISFIES(XW,XW EQ 'BOX)
    -> EXISTS(BOX) & ADDAV(BOX,O) & ISA(BOX,'BOX) & CUROBJ(BOX,P)
       & REFERS(BOX,BOX) & ERRREF(BOX,X) & NEWOBJ(BOX) & NEGATE(1);
N51: "ADD AVN" = ADDAV(O,OP) & NEWAV(OP,A,V,I) -> HASAV(O,A,V,I) & NEGATE(1,2);
END;


                       ------------------------


       % F = FIND REFERENTS, B = BACKUP REFERENTS %        % PAGE 4 %

EXPR MILFB(); BEGIN

F1: "QWORD FIND" = QWFIND(O,X) & ISA(O2,N) -> FINDPOSS(O,O2) & NEGATE(1);
F2: "QWORD NFND" = QWFIND(O,X) & NOT( EXISTS(O2,N) & ISA(O2,N) )
    -> ERROR(X,'(NO OBJECTS)) & NEGATE(1);
F5: "DEF FIND" = DEFFND(O,X) & ISA(O2,N) & NOT NEWOBJ(O2)
    -> FINDPOSS(O,O2) & NEGATE(1);
F6: "DEF NFND" = DEFFND(O,X) & NOT( EXISTS(O2,N) & ISA(O2,N) )
    -> ERROR(X,'(NO OBJECTS)) & NEGATE(1);

F11: "OBJ REF NUL" = OCHK(O,X) & NOT( EXISTS(O2) & FINDPOSS(O,O2) )
    -> NULLREF(O,X) & NEGATE(1);
F13: "OBJ FND" = OCHK(O,X) & FINDPOSS(O,O2)
       & NOT( EXISTS(O3) & FINDPOSS(O,O3) & VNEQ(O2,O3) )
    -> REFERS(O,O2) & TRACING(TRACEPRINTM(<O,'REFERS,O2>)) & NEGATE(1,2);
F15: "OBJ MULT" = OCHK(O,X) & FINDPOSS(O,O2) & FINDPOSS(O,O3) & VNEQ(O2,O3)
       & SATISFIES2(O2,O3,O2 LEXORDER O3)
       & NOT( EXISTS(O4) & FINDPOSS(O,O4) & VNEQ(O4,O3) & VNEQ(O4,O2)
              & SATISFIES2(O4,O3,O4 LEXORDER O3) )    % MAKES FIRE UNIQUE %
    -> TRACING(TRACEPRINTM(<O,'AMBIG,X,O2,O3,"..">)) & NEGATE(1);

F21: "N RESTR" = NRESTR(O,X,XW) & FINDPOSS(O,O2) & NOT ISA(O2,XW)
    -> OCHK(O,X) & NEGATE(ALL);
F23: "N INCON" = NRESTR(O,X,XW) & REFERS(O,OA) & NOT ISA(OA,XW)
    -> NULLREF(O,X) & NEGATE(ALL);
F27: "AV RESTR" = AVRESTR(O,X,A,V,I) & FINDPOSS(O,O2) & NOT HASAV(O2,A,V,I)
    -> OCHK(O,X) & NEGATE(2);
F29: "AV INCON" = AVRESTR(O,X,A,V,I) & REFERS(O,OA) & NOT HASAV(OA,A,V,I)
    -> NULLREF(O,X) & NEGATE(ALL);

F31: "REL RESTR" = RELRESTR(O,X,R,O2,I) & FINDPOSS(O,O3)
       & NOT HASREL(O3,R,O2,I)
    -> OCHK(O,X) & NEGATE(2);
F35: "PRED RESTR" = PREDRESTR(O,X,A,V,I) & FINDPOSS(O,O2) & NOT HASAV(O2,A,V,I)
    -> OCHK(O,X) & NEGATE(2);
F41: "PRED RESTR" = ISPRED(X) & CUROBJ(O,P) & ISAV(X,A,V,I) & NOT OLDAV(X)
    -> PREDRESTRCHK(O,X,A,V,I) & OLDAV(X);

F51: "NULL REF ERR" = NULLREF(O,X) & CUROBJ(O,P) & NOT SATISFIES(P,P EQ 'MAIN)
    -> ERROR(X,'(NO SUCH) );
F53: "NULL REF ERR" = NULLREF(O,X) & CUROBJ(O,P) & SATISFIES(P,P EQ 'MAIN)
       & SENTENCE(S) & NOT GSQE(S) & NOT GSQW(S)
    -> ERROR(X,'(NO SUCH) );


                       ------------------------


       % B = BACKUP REFERENTS %                          % PAGE 5 %

B1: "DEF REF" = REFERS(O,OA) & CUROBJ(O,P) & HASRELN(P,R,I) & ERRREF(P,X)
       & NOT OLDREF(O)
```

```
   -> RELRESTRCHK(P,X,R,DA,S) & CUROBJP(0,P) & OLDREF(0);
B9: "DEF REF" = REFERS(0,DA) & CUROBJP(0,P) & HASRELNP(P,R,S) & ERRREF(P,X)
      & NOT OLDREF(0)
   -> RELRESTRCHK(P,X,R,DA,S) & OLDREF(0);

B11: "REL RCHK NEW" = RELRESTRCHK(0,X,R,D2,S) & NEWOBJ(0)
      & NOT HASREL(0,R,D2,S)
      & NOT( EXISTS(N) & HASREL(0,R,D2,N) & VNEQ(N,S) )
   -> HASREL(0,R,D2,S) & NEGATE(1);
B13: "REL RCHK EX" = RELRESTRCHK(0,X,R,D2,S) & FINDPOSS(0,DA)
      & HASREL(0A,R,D2,S)
   -> RELRESTRT(0,X,R,D2,S) & NEGATE(1);
B14: "REL RCHK QW" = RELRESTRCHK(0,X,R,D2,S) & GSQW(SN) & CUROBJ(0,P)
      & SATISFIES(P,P EQ 'MAIN)
   -> RELRESTRT(0,X,R,D2,S) & NEGATE(1);
B15: "REL RCHK RED" = RELRESTRCHK(0,X,R,D2,S) & REFERS(0,DA)
      & HASREL(0A,R,D2,S)
   -> RELREDUNT(0,X,R,D2,S) & NEGATE(1);
B17: "REL RCHK ERR" = RELRESTRCHK(0,X,R,D2,S) & FINDPOSS(0,DX)
      & NOT( EXISTS(0A) & FINDPOSS(0,DA) & HASREL(0A,R,D2,S) )
      & SENTENCE(SN) & NOT( EXISTS(P) & GSQW(SN) & CUROBJ(0,P)
            & SATISFIES(P,P EQ 'MAIN) )
   -> ERROR(X,'(WHICH ONE ??)) & NEGATE(1);
B18: "REL RCHK INC" = RELRESTRCHK(0,X,R,D2,S) & NOT NEWOBJ(0) & REFERS(0,DA)
      & NOT HASREL(0A,R,D2,S)
   -> RELINCONT(0,X,R,D2,S) & NEGATE(1);
B19: "REL RCHK INC-" = RELRESTRCHK(0,X,R,D2,S) & REFERS(0,DA)
      & HASREL(0A,R,D2,N) & VNEQ(N,S)
   -> RELINCONT(0,X,R,D2,S) & NEGATE(1);

B21: "PRED RCHK NEW" = PREDRESTRCHK(0,X,A,V,S) & NEWOBJ(0) & NOT HASAV(0,A,V,S)
      & NOT( EXISTS(N) & HASAV(0,A,V,N) & VNEQ(N,S) )
   -> HASAV(0,A,V,S) & NEGATE(1);
B23: "PRED RCHK EX" = PREDRESTRCHK(0,X,A,V,S) & FINDPOSS(0,DA)
      & HASAV(0A,A,V,S)
   -> PREDRESTRT(0,X,A,V,S) & NEGATE(1);
B24: "PRED RCHK QW" = PREDRESTRCHK(0,X,A,V,S) & GSQW(SN) & CUROBJ(0,P)
      & SATISFIES(P,P EQ 'MAIN)
   -> PREDRESTRT(0,X,A,V,S) & NEGATE(1);
B26: "PRED RCHK RED" = PREDRESTRCHK(0,X,A,V,S) & REFERS(0,DA) & HASAV(0A,A,V,S)
   -> PREDREDUNT(0,X,A,V,S) & NEGATE(1);
B27: "PRED RCHK ERR" = PREDRESTRCHK(0,X,A,V,S) & FINDPOSS(0,DX)
      & NOT( EXISTS(0A) & FINDPOSS(0,DA) & HASAV(0A,A,V,S) )
      & SENTENCE(SN) & NOT( EXISTS(P) & GSQW(SN) & CUROBJ(0,P)
            & SATISFIES(P,P EQ 'MAIN) )
   -> ERROR(X,'(WHICH ONE ??)) & NEGATE(1);
B28: "PRED RCHK INC" = PREDRESTRCHK(0,X,A,V,S) & NOT NEWOBJ(0) & REFERS(0,DA)
      & NOT HASAV(0A,A,V,S)
   -> PREDINCONT(0,X,A,V,S) & NEGATE(1);
B29: "PRED RCHK INC-" = PREDRESTRCHK(0,X,A,V,S) & REFERS(0,DA)
      & HASAV(0A,A,V,N) & VNEQ(N,S)
   -> PREDINCONT(0,X,A,V,S) & NEGATE(1);

B31: "BK REL REDUN" = RELREDUN(0,X,R,D2,S) & FINDPOSS(03,D4)
      & HASREL(04,R,D7,S)
   -> FINDAMBIGR(0,X,R,D7,S,0);
B33: "F AMB REL" = FINDAMBIGR(0,X,R,D2,S,1) & CUROBJP(0,P) & FINDPOSS(P,DA)
      & HASREL(0A,R,D2,S) & REFERS(04,D2) & CUROBJP(04,D3) & CUROBJ(04,D3)
   -> RELRESTRT(P,X,R,D7,S) & CUROBJ(04,P) & CUROBJP(04,P) & NEGATE(1,2,6,7)
      & NOT RELREDUN(1,X,R,D2,S);
B34: "F AMB REL" = FINDAMBIGR(0,X,R,D2,S,1) & CUROBJP(0,P) & FINDPOSS(P,DA)
      & HASREL(0A,R,D2,S) & REFERS(04,D7) & CUROBJP(04,D3) & NOT CUROBJ(04,D3)
   -> RELRESTRT(P,X,R,D7,S) & CUROBJP(04,P) & NEGATE(1,2,6)
      & NOT RELREDUN(1,X,R,D7,S);
B35: "F AMB BK" = FINDAMBIGR(0,X,R,D2,S,1) & CUROBJP(0,P)
      & NOT( EXISTS(03) & FINDPOSS(P,D3) )
   -> FINDAMBIGR(P,X,R,D2,S,1) & NEGATE(1);
B36: "F AMB -" = FINDAMBIGR(0,X,R,D2,S,1) & CUROBJP(0,P) & FINDPOSS(P,DA)
      & NOT HASREL(0A,R,D7,S)
      & NOT( EXISTS(03) & FINDPOSS(P,D3) & HASREL(03,R,D2,S) )
   -> NEGATE(1);
B38: "BK REL INCON" = RELINCON(0,X,R,D7,S) & CUROBJ(03,D) & CUROBJP(0,P)
      & NOT SATISFIES(P,P EQ 'MAIN)
   -> RELRESTRCHK(P,X,R,D7,S) & CUROBJ(03,P) & NEGATE(ALL);
B39: "BK REL INCON-C" = RELINCON(0,X,R,D2,S) & CUROBJP(0,P)
      & NOT SATISFIES(P,P EQ 'MAIN) & CUROBJP(03,D)
      & NOT( EXISTS(04) & CUROBJ(04,D) )
   -> RELRESTRCHK(P,X,R,D2,S) & CUROBJP(03,P) & NEGATE(ALL);

B41: "BK PRED REDUN" = PREDREDUN(0,X,A,V,S) & FINDPOSS(03,D4) & HASAV(04,A,V,S)
   -> FINDAMBIGP(0,X,A,V,S,0);
B43: "F AMB PRED" = FINDAMBIGP(0,X,A,V,S,1) & CUROBJP(0,P) & FINDPOSS(P,DA)
      & HASAV(0A,A,V,S) & CUROBJP(1,D2) & CUROBJ(1,D2)
   -> PREDRESTRT(P,X,A,V,S) & CUROBJ(1,P) & CUROBJP(1,P) & NEGATE(1,2,5,6)
      & NOT PREDREDUN(1,X,A,V,S);
B44: "F AMB PRED" = FINDAMBIGP(0,X,A,V,S,1) & CUROBJP(0,P) & FINDPOSS(P,DA)
      & HASAV(0A,A,V,S) & CUROBJP(1,D2) & NOT CUROBJ(1,D2)
   -> PREDRESTRT(P,X,A,V,S) & CUROBJP(1,P) & NEGATE(1,2,5)
      & NOT PREDREDUN(1,X,A,V,S);
B45: "F AMB BK" = FINDAMBIGP(0,X,A,V,S,1) & CUROBJP(0,P)
      & NOT( EXISTS(02) & FINDPOSS(P,D2) )
   -> FINDAMBIGP(P,X,A,V,S,1) & NEGATE(1);
B46: "F AMB -" = FINDAMBIGP(0,X,A,V,S,1) & CUROBJP(0,P) & FINDPOSS(P,DA)
      & NOT HASAV(0A,A,V,S)
      & NOT( EXISTS(02) & FINDPOSS(P,D2) & HASAV(02,A,V,S) )
   -> NEGATE(1);
B48: "BK PRED INCON" = PREDINCON(0,X,A,V,S) & CUROBJ(0,P) & CUROBJP(P,PP)
   -> PREDRESTRCHK(P,X,A,V,S) & CUROBJP(P,PP) & NEGATE(1,2) & NOT CUROBJP(0,P);
      % NO B49 NECESSARY, TO PARALLEL B39, BECAUSE PRED FORCES CUROBJ %

B51: "NPBND UNDO" = NPBOUND(X) & CUROBJ(0,P) & REFERS(0,DA)
      & SATISFIES(P,P NEQ 'MAIN)
   -> NEGATE(2);
B53: "NPBND UNDOP" = NPBOUND(X) & CUROBJP(0,P) & REFERS(0,DA)
      & SATISFIES(P,P NEQ 'MAIN)
   -> NEGATE(2);
B55: "NPBND REDO" = NPBOUND(X) & NOT( EXISTS(0,DX) & FINDPOSS(0,DX)
            & NOT( EXISTS(SN,P) & GSQW(SN) & CUROBJP(0,P)
                  & SATISFIES(P,P EQ 'MAIN) ) )
      & CUROBJP(0,P) & SATISFIES(P,P EQ 'MAIN)
   -> CUROBJ(0,P);
B57: "NPBND ERR" = NPBOUND(X) & FINDPOSS(0,DX) & ERRREF(0,L)
      & NOT( EXISTS(SN,P) & GSQW(SN) & CUROBJP(0,P)
            & SATISFIES(P,P EQ 'MAIN) )
   -> ERROR(L,'(WHICH ONE ??));
B58: "NPBND DEL" = NPBOUNDL(W) -> NOT NPBOUND(W) & NEGATE(1);

END;
```

```
   % M - SEMANTIC CASES FOR DIFFERENT SENTENCE TYPES %     % PAGE 8 %

EXPR MILMQ; BEGIN

M1: "REL INCON SD" = RELINCON(0,X,R,D2,S) & GSD(SN) & CUROBJP(0,P)
      & SATISFIES(P,P EQ 'MAIN) & REFERS(0,DA)
      & NOT( EXISTS(03,D4,D5) & CUROBJP(03,D4) & FINDPOSS(03,D5) )
   -> HASREL(0A,R,D2,S) & NEGATE(1);
M2: "PRED INCON SD" = PREDINCON(0,X,A,V,S) & GSD(SN) & CUROBJP(0,P)
      & SATISFIES(P,P EQ 'MAIN) & REFERS(0,DA)
      & NOT( EXISTS(02,D3,D4) & CUROBJP(02,D3) & FINDPOSS(02,D4) )
   -> HASAV(0A,A,V,S) & NEGATE(1);
M5: "PRED REDUN SD" = PREDREDUN(0,X,A,V,S) & GSD(SN) & CUROBJP(01,P)
      & SATISFIES(P,P EQ 'MAIN) & REFERS(01,DA)
      & NOT( EXISTS(02,D3,D4) & CUROBJP(02,D3) & FINDPOSS(02,D4) )
   -> HASAV(0A,A,V,S) & NEGATE(1);

M11: "ANSREL I" = RELINCON(0,X,R,D2,S) & CUROBJP(0,P) & SATISFIES(P,P EQ 'MAIN)
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSE(SN) & NOT GSQW(SN)
      & NOT GSQWR(SN)        % THAT LEAVES GSQE OR GSQD %
   -> ANSRELINC(0,X,R,D2,S) & NEGATE(1);
M12: "ANSREL R" = RELREDUN(0,X,R,D2,S) & CUROBJP(03,P)
      & SATISFIES(P,P EQ 'MAIN)
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSE(SN) & NOT GSQW(SN)
      & NOT GSQWR(SN)
      & NOT( EXISTS(04,D5) & FINDPOSS(04,D5) & HASREL(05,R,D2,S) )
      % JUST IN CASE RELRED IS THE Q BEING ASKED; ANS WILL BE NO %
   -> ANSRELRED(03,R,D2,S) & NEGATE(1);
M15: "ANSPRED I" = PREDINCON(0,X,A,V,S) & CUROBJ(0,P)
      & SATISFIES(P,P EQ 'MAIN) & GSQE(SN)
   -> ANSPRED(0,A,V,S) & NEGATE(1);
M16: "ANSPRED R" = PREDREDUN(0,X,A,V,S) & CUROBJP(02,P)
      & SATISFIES(P,P EQ 'MAIN) & GSQE(SN)
      & NOT( EXISTS(03,D4) & FINDPOSS(03,D4) & HASAV(04,A,V,S) )
   -> ANSPREDRED(02,A,V,S) & NEGATE(1);

M51: "REL INCON ERR" = RELINCON(0,X,R,D2,S) & CUROBJP(0,P)
      & SATISFIES(P,P EQ 'MAIN)       % THIS IS FOR GSE, GSQW, GSQWR %
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSQE(SN) & NOT GSQD(SN)
   -> ERROR(X,'(INCONSISTENT) );
M53: "PRED INCON ERR" = PREDINCON(0,X,A,V,S) & CUROBJP(0,P)
      & SATISFIES(P,P EQ 'MAIN)       % THIS IS FOR GSE, GSQW, GSQWR %
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSQE(SN) & NOT GSQD(SN)
```

```
  -> ERROR(X,'(INCONSISTENT) );

END;


                    --------------------


    % V = REPLY, D = DESCRIBE %                    % PAGE 7 %

EXPR MILVD(); BEGIN

V2: "REPLY SD" = SENTBOUND(S) & GSD(S) -> REPLY('(OKAY));
V8: "REPLY QUIT" = REPLY(B) & SCANFIN(X) -> NEGATE(2);
V10: "REPLY SQW1" = SENTBOUND(S) & GSQW(S) & CUROBJ(O,P) & REFERS(O,OA)
       & SATISFIES(P,P EQ 'MAIN)
   -> DESCRIBE(OA) & QWREPLY(OA);
V12: "REPLY SQWO" = SENTBOUND(S) & GSQW(S) & NULLREF(O,X) & CUROBJ(O,P)
       & SATISFIES(P,P EQ 'MAIN)
   -> REPLY('(NOTHING));
V14: "REPLY SQWM" = SENTBOUND(S) & GSQW(S) & CUROBJ(O,P) & FINDPOSS(O,OX)
       & SATISFIES(P,P EQ 'MAIN)
   -> DESCRIBE(OX) & QWREPLY(OX);
V15: "REPLY SQWP" = QWREPLY(X) & DESCRPHRASE(X,L) -> REPLY(L) & NEGATE(ALL);

V17: "REPLY SQWR1" = SENTBOUND(SO) & GSQWR(SO) & CUROBJ(O,P) & REFERS(O,OA)
       & SATISFIES(P,P EQ 'MAIN) & HASREL(OA,R,O2,S)
   -> QWRDESCR2(OA) & DESCRIBE(OA) & DESCRIBE(O2) & QWRREPLY1(OA,R,O2,S);
V18: "REPLY SQWR1+" = QWRDESCR2(O1) & HASREL(O2,R,O1,S)
   -> DESCRIBE(O2) & QWRREPLY2(O1,O2,R,S) & NEGATE(1);
V19: "REPLY SQWRO" = SENTBOUND(SO) & GSQWR(SO) & CUROBJ(O,P)
       & SATISFIES(P,P EQ 'MAIN) & REFERS(O,OA)
       & NOT( EXISTS(O2,R,S) & HASREL(OA,R,O2,S) )
   -> QWRDESCR2(OA) & DESCRIBE(OA) & QWRREPLY3(OA);

V20: "REPLY SE" = SENTBOUND(S) & GSE(S) -> REPLY('(OKAY));
V25: "REPLY SQE NUL" = NULLREF(O,X) & CUROBJ(O,P) & SATISFIES(P,P EQ 'MAIN)
       & GSQE(S)
   -> REPLY('(NO) );
V30: "REPLY SQDE REL+" = SENTBOUND(SM) & ANSREL(O,R,O2,S) & REFERS(O,OA)
       & HASREL(OA,R,O2,S)
   -> REPLY('(YES)).
V31: "REPLY SQDE REL-" = SENTBOUND(SM) & ANSREL(O,R,O2,N) & REFERS(O,OA)
       & HASREL(OA,R,O2,P) & VNEQ(N,P)
   -> REPLY('(NO));
V32: "REPLY SQDE RELU" = SENTBOUND(SM) & ANSREL(O,R,O2,P) & REFERS(O,OA)
       & NOT( EXISTS(N) & HASREL(OA,R,O2,N) )
   -> REPLY('(NO INFORMATION ON RELATION) @ <R>);
V35: "REPLY SQDE PRED+" = SENTBOUND(SM) & ANSPRED(O,A,V,S) & REFERS(O,OA)
       & HASAV(OA,A,V,S)
   -> REPLY('(YES));
V36: "REPLY SQDE PRED-" = SENTBOUND(SM) & ANSPRED(O,A,V,N) & REFERS(O,OA)
       & HASAV(OA,A,V,P) & VNEQ(N,P)
   -> REPLY('(NO));
V37: "REPLY SQDE PREDU" = SENTBOUND(SM) & ANSPRED(O,A,V,P) & REFERS(O,OA)
       & NOT( EXISTS(N) & HASAV(OA,A,V,N) )
   -> REPLY('(NO INFORMATION ON) @ <A,V>);

V40: "ANS REL INC" = ANSRELINC(O,X,R,O2,S) & SENTBOUND(SM) & ANSPREDFIN(A,V,S2)
   -> ERROR(X,'(INCONSISTENT));
V42: "ANS REL OK1" = ANSRELINC(O,X,R,O2,S) & SENTBOUND(SM)
       & NOT( EXISTS(A,V,S2) & ANSPREDFIN(A,V,S2) )
   -> ANSREL(O,R,O2,S);
V44: "ANS REL RED" = ANSRELRED(O,R,O2,S) & SENTBOUND(SM) & ANSPREDFIN(A,V,S2)
   -> ANSPREDFIN(A,V,S2) & NEGATE(1);
V46: "ANS REL OKR" = ANSRELRED(O,R,O2,S) & SENTBOUND(SM)
       & NOT( EXISTS(A,V,S2) & ANSPREDFIN(A,V,S2) )
       & NOT( EXISTS(O3,R2,O4,S2,X2) & ANSRELINC(O3,X2,R2,O4,S2) )
       & NOT( EXISTS(O3,A,V,S2) & ANSPRED(O3,A,V,S2) )
   -> ANSREL(O,R,O2,S);
V48: "ANS PRED FIN" = ANSPREDFIN(A,V,S) & SENTBOUND(SM)
       & NOT( EXISTS(O,R,O2,S2) & ANSRELRED(O,R,O2,S2) )
       & NOT( EXISTS(O,X,R,O2,S2) & ANSRELINC(O,X,R,O2,S) )
       & CUROBJP(O,P) & SATISFIES(P,P EQ 'MAIN)
   -> ANSPRED(O,A,V,S) & NEGATE(1);
V49: "ANS PRED RED" = ANSPREDRED(OM,A,V,S) & SENTBOUND(SM)
       & NOT( EXISTS(A2,V2,S2,O2) & ANSPRED(O2,A2,V2,S2) )
       & NOT( EXISTS(O,R,O2,X,S2) & ANSRELINC(O,X,R,O2,S2) )
       & NOT( EXISTS(O,R,O2,S2) & ANSREL(O,R,O2,S2) )
   -> ANSPRED(OM,A,V,S);

D1: "DESCRIBE" = DESCRIBE(X)
   -> DESCRAV(X,'SIZE,'POS,'(THE)) & DESCRMX('SIZE,'COLOR)
       & DESCRMX('COLOR,'ISA) & NEGATE(1);
D8: "DESCR NEXT" = DESCRAV(X,A,S,L) & SATISFIES(S,S EQ 'POS)
       & NOT( EXISTS(V2) & HASAV(X,A,V2,S) & NOT DESCRIBED(X,A,V2,S) )
   -> DESCRAV(X,A,'NEG,L) & NEGATE(1);
D9: "DESCR NEXT" = DESCRAV(X,A,S,L) & SATISFIES(S,S EQ 'NEG) & DESCRMX(A,AN)
       & NOT SATISFIES(AN,AN EQ 'ISA)
       & NOT( EXISTS(V2) & HASAV(X,A,V2,S) & NOT DESCRIBED(X,A,V2,S) )
   -> DESCRAV(X,AN,'POS,L) & NEGATE(1);
D10: "DESCR ISA" = DESCRAV(X,A,S,L) & SATISFIES(S,S EQ 'NEG) & DESCRMX(A,AN)
       & SATISFIES(AN,AN EQ 'ISA)
       & NOT( EXISTS(V2) & HASAV(X,A,V2,S) & NOT DESCRIBED(X,A,V2,S) )
       & ISA(X,N)
   -> DESCRPHRASE(X,L @ <N>) & NEGATE(1);
D11: "DESCR AV POS" = DESCRAV(X,A,S,L) & SATISFIES(S,S EQ 'POS)
       & HASAV(X,A,V,S) & NOT DESCRIBED(X,A,V,S)
       & NOT( EXISTS(V2) & HASAV(X,A,V2,S) & VNEQ(V,V2)
            & NOT DESCRIBED(X,A,V2,S)& SATISFIES2(V,V2,V2 LEXORDER V) )
   -> DESCRAV(X,A,S,L @ <V>) & DESCRIBED(X,A,V,S) & NEGATE(1);
D12: "DESCR AV NEG" = DESCRAV(X,A,S,L) & SATISFIES(S,S EQ 'NEG)
       & HASAV(X,A,V,S) & NOT DESCRIBED(X,A,V,S)
       & NOT( EXISTS(V2) & HASAV(X,A,V2,S) & VNEQ(V,V2)
            & NOT DESCRIBED(X,A,V2,S) & SATISFIES2(V,V2,V2 LEXORDER V) )
   -> DESCRAV(X,A,S,L @ <UN-,V>) & DESCRIBED(X,A,V,S) & NEGATE(1);

D21: "DESCR REL INIT" = QWRREPLY1(O1,R,O2,S) & DESCRPHRASE(O1,X)
       & NOT( EXISTS(L,Z) & QWRPHRASE1(O1,L,Z) )
       & NOT( EXISTS(O3,S2,R2) & QWRREPLY1(O1,R2,O3,S2)
            & SATISFIES2(O2,O3,O3 LEXORDER O2) & VNEQ(O2,O3) )
       % MAY BE REDUNDANT FIRINGS IF SAME OBJECT RELATED SEVERAL WAYS %
   -> QWRPHRASE1(O1,X,'IS);
D22: "DESCR REL POS" = QWRPHRASE1(O1,L,Z) & QWRREPLY1(O1,R,O,S)
       & SATISFIES(S,S EQ 'POS) & DESCRPHRASE(O,X)
       & NOT( EXISTS(R2,O2,S2) & QWRREPLY1(O1,R2,O2,S2) & VNEQ(O,O2)
            & SATISFIES2(O2,O,O2 LEXORDER O) )
       & NOT( EXISTS(R2,S2) & QWRREPLY1(O1,R2,O,S2) & VNEQ(R,R2)
            & SATISFIES2(R2,R,R2 LEXORDER R) )
   -> QWRPHRASE1(O1,L @ <Z,R> @ X,'AND) & NEGATE(1,2);
D23: "DESCR REL NEG" = QWRPHRASE1(O1,L,Z) & QWRREPLY1(O1,R,O,S)
       & SATISFIES(S,S EQ 'NEG) & DESCRPHRASE(O,X)
       & NOT( EXISTS(R2,O2,S2) & QWRREPLY1(O1,R2,O2,S2) & VNEQ(O,O2)
            & SATISFIES2(O2,O,O2 LEXORDER O) )
       & NOT( EXISTS(R2,S2) & QWRREPLY1(O1,R2,O,S2) & VNEQ(R,R2)
            & SATISFIES2(R2,R,R2 LEXORDER R) )
   -> QWRPHRASE1(O1,L @ <Z,'NOT,R> @ X,'AND) & NEGATE(1,2);
D24: "DESCR REL-" = QWRPHRASE1(O1,L,Z)
       & NOT( EXISTS(R,O,S) & QWRREPLY1(O1,R,O,S) )
   -> REPLY(L) & NEGATE(1);

D25: "DESCR REL= INIT" = QWRREPLY2(O1,O,R,S) & DESCRPHRASE(O,X)
       & NOT( EXISTS(L,Z) & QWRPHRASE2(O1,L,Z) )
       % MAY BE REDUNDANT FIRINGS IF SAME OBJECT RELATED SEVERAL WAYS %
   -> QWRPHRASE2(O,X,'IS);
D26: "DESCR REL= POS" = QWRPHRASE2(O1,L,Z) & QWRREPLY2(O1,O,R,S)
       & SATISFIES(S,S EQ 'POS)
       & NOT( EXISTS(R2,S2) & QWRREPLY2(O1,O,R2,S2) & VNEQ(R,R2)
            & SATISFIES2(R,R2,R2 LEXORDER R) )
   -> QWRPHRASE2(O1,L @ <Z,R,'IT>,'AND) & NEGATE(1,2);
D27: "DESCR REL= NEG" = QWRPHRASE2(O1,L,Z) & QWRREPLY2(O1,O,R,S)
       & SATISFIES(S,S EQ 'NEG)
       & NOT( EXISTS(R2,S2) & QWRREPLY2(O1,O,R2,S2) & VNEQ(R,R2)
            & SATISFIES2(R,R2,R2 LEXORDER R) )
   -> QWRPHRASE2(O1,L @ <Z,'NOT,R,'IT>,'AND) & NEGATE(1,2);
D28: "DESCR REL= -" = QWRPHRASE2(O1,L,Z)
       & NOT( EXISTS(R,O1,S) & QWRREPLY2(O1,O,R,S) )
   -> REPLY(L) & NEGATE(1);

D29: "DESCR REL3" = QWRREPLY3(O) & DESCRPHRASE(O,X)
   -> REPLY(X @ '(IS NOWHERE)) & NEGATE(1);

END;


                    --------------------


    % X = EXAMPLES %                    % PAGE 8 %
EXPR MILX(); BEGIN           PSMACRO(MILING);

X1: TEST1(P) -> SAVQ(1,'(A LARGE GREEN BLOCK IS ON A RED TABLE));
X2: TEST2(P) -> SAVQ(2,'(A BLUE BALL IS ON THE TABLE));
X3: TEST3(P) -> SAVQ(3,'(THE BALL IS NEAR THE BLOCK));
X4: TEST4(P) -> SAVQ(4,'(A BLUE BALL IS ON THE BLOCK));
X5: TEST5(P) -> SAVQ(5,'(THE BALL ON THE BLOCK IS SMALL));
```

```
   END;

EXPR MILX2(); BEGIN           PSMACRO(MILIN);

X6; TEST6(P) -> SAYQ(6,'(WHAT IS ON THE BLOCK) );
X7; TEST7(P) -> SAYQ(7,'(WHAT IS BLUE) );
X8; TEST8(P) -> SAYQ(8,'(THERE IS A BOX ON THE TABLE) );
X9; TEST9(P) -> SAYQ(9,'(IS THE BOX ON THE TABLE) );
X10; TEST10(P) -> SAYQ(10,'(IS THE BOX ON THE TABLE NEAR THE BLOCK) );

   END;

EXPR MILX3(); BEGIN           PSMACRO(MILIN);

X11; TEST11(P) -> SAYQ(11,'(THERE IS A BOX ON A RED FLOOR WHICH IS NOT RED) );
X12; TEST12(P) -> SAYQ(12,'(WHAT IS NOT BLUE) );
X13; TEST13(P) -> SAYQ(13,'(THE BOX THAT IS NOT RED IS NOT ON THE TABLE) );
X14; TEST14(P) -> SAYQ(14,'(WHAT IS NOT ON THE TABLE) );
X15; TEST15(P) -> SAYQ(15,'(IS THERE A BLACK BOX ON THE FLOOR) );

   END;

EXPR MILX4(); BEGIN           PSMACRO(MILIN);

X16; TEST16(P) -> SAYQ(16,'(WHAT THAT IS NOT RED IS ON THE FLOOR) );
X17; TEST17(P) -> SAYQ(17,'(WHAT IS) );
X18; TEST18(P) -> SAYQ(18,'(A SMALL RED BALL IS IN THE BOX ON THE RED FLOOR) );
X19; TEST19(P) -> SAYQ(19,'(THERE IS A LARGE GREEN BALL IN THE BOX ON THE
          FLOOR NEAR THE BALL IN THE BOX ON THE FLOOR) );

   END;

EXPR MILX5(); BEGIN           PSMACRO(MILIN);

X20; TEST20(P) -> SAYQ(70,'(WHERE IS THE BOX THAT IS NOT RED) );
X21; TEST21(P) -> SAYQ(71,'(WHERE IS THE BALL IN THE BOX ON THE RED FLOOR
                    THAT IS LARGE) );
X22; TEST22(P) -> SAYQ(22,'(WHERE IS THE BALL IN THE BOX ON THE RED FLOOR
                    THAT IS RED) );
X23; TEST23(P) -> SAYQ(23,'(THERE IS A BLACK BALL NEAR THE GREEN BALL THAT IS
                    NOT IN THE BOX ON THE FLOOR) );
X24; TEST24(P) -> SAYQ(24,'(THE RED BALL IS NEAR THE GREEN BALL) );
X25; TEST25(P) -> SAYQ(25,'(IS THE BALL NEAR THE GREEN BALL IN THE BOX THAT
                    IS NOT ON THE RED TABLE BLACK) );

   END;

$     REMAINS TO DO :
    UN-CREATION OF CREATED OBJECTS, IF ERROR
    "WHERE IS EVERYTHING" NOT IN AT ALL (MUST GET PAST B57)
          - SHOULD BE TYPE GSQWRE, SINCE DON'T WANT CONVERSES AS IN QWR
    A EQUIV ANY IN QUESTIONS
    CONVERSE RELATIONS, OR <NOUN> ON WHICH <NP> IS (NOT)
    WHAT <ATTRIB> IS <NP>
    CHANGE GRAMMAR TO ANTICIPATORY ?
    CONJUNCTIONS (AND, OR) ANYWHERE
    ABSENCE OF REL OR AV -> NOT; CLASS EXCLUSIONS FOR NEGATIVE AV'S
    GENERAL DATABASE CONSISTENCY; INCL CLASS EXCLUSIONS
    MUTUAL DISAMBIGUATION: TWO RELATED NOUNS BY THEIR RELATION
          DETERMINE EACH OTHER
    $
END.
```

Appendix B. CROSS-REFERENCE OF MILIPS PREDICATES

```
INDEX OF MILIPS PREDS

ADDAV
 LHSUSES N51
 RHSUSES N41 N42 N43 N44 N45 -N51
ANSPRED
 LHSUSES V35 V36 V37
 NESTEDL V46 V49
 RHSUSES M15 V46 V49
ANSPREDFIN
 LHSUSES V40 V44 V48
 NESTEDL V42 V46
 RHSUSES A14 V44 -V48
ANSPREDRED
 LHSUSES V49
 RHSUSES M16
ANSREL
 LHSUSES V30 V31 V32
 NESTEDL V49
 RHSUSES V42 V46
ANSRELINC
 LHSUSES V40 V42
 NESTEDL V46 V48 V49
 RHSUSES M11
ANSRELRED
 LHSUSES V44 V46
 NESTEDL V48
 RHSUSES M12 -V44
AVRESTR
 LHSUSES F27 F29
 RHSUSES A1 -F29
COPSIGN
 LHSUSES R11
 NESTEDL R12
 RHSUSES G31 -G31 G32 -G32 -R11
CUROBJ
 LHSUSES A1 A5 R11 R12 N1 N5 N5A N31 N33 F41 F51 F52 B1 B14 B20 B33 -B34 B38
 B43 -B44 B48 B51 M15 V10 V12 V14 V17 V19 V25
 NESTEDL N2 N8 N10 B17 B27 B39
 RHSUSES G13 N1 -N1 N2 N5 -N5 N6 -N31 M41 N42 N43 N44 N45 B33 -B33 B38 -B38
 B43 -B43 B48 -B48 -B51 B55
CUROBJP
 LHSUSES B3 B33 B34 B35 B36 B38 B39 B43 B44 B45 B46 B48 B53 B55 M1 M2 M5 M11
 M12 M16 M51 M53 V48
 NESTEDL B55 B57 M1 M2 M5
 RHSUSES G13 N1 N2 N5 B1 B33 -B33 B34 -B34 -B38 B39 -B39 B43 -B43 B44 -B44
 -B48 -B53
DEFDET
 LHSUSES N1 N2 N22 -N29
 RHSUSES G1 G2 G5
DEFFND
 LHSUSES F5 F6
 RHSUSES N1 N2 -F5 -F6
DESCRAV
 LHSUSES D2 D3 D4 D11 D12
 RHSUSES D1 D2 -D2 D3 -D3 -D4 D11 -D11 D12 -D12
DESCRIBE
 LHSUSES D1
 RHSUSES V10 V14 V17 V18 V19 -D1
DESCRIBED
 LHSUSES -D11 -D12
 NESTEDL -D2 -D3 -D4 -D11 -D12
 RHSUSES D11 D12
DESCRNOUN
 LHSUSES D3 D4
 RHSUSES D1
DESCRPHRASE
 LHSUSES V15 D21 D22 D23 D25 D26
 RHSUSES -V15 D4
DETSEEN
 LHSUSES A19 -A25 -N1 -N5
 RHSUSES N1 N2 N5 N6
ENDMARK
 LHSUSES S0 -S1 S4 T57 -E4 E6 A14 N10 -N10
 NESTEDL A25
EQA
 LHSUSES G5 G6 G7
 RHSUSES -G5 -G6 -G7
EQNALL
 LHSUSES T61
```

```
  RHSUSES -T61
EQBLACK
  LHSUSES T16
  RHSUSES -T16
EQBLOCK
  LHSUSES T44
  RHSUSES -T44
EQBLUE
  LHSUSES T13
  RHSUSES -T13
EQBOX
  LHSUSES T53
  RHSUSES -T53
EQFLOOR
  LHSUSES T50
  RHSUSES -T50
EQGREEN
  LHSUSES T10
  RHSUSES -T10
EQIN
  LHSUSES T31
  RHSUSES -T31
EQIS
  LHSUSES T1 T2 T6
  RHSUSES -T1 -T6
EQLARGE
  LHSUSES T21
  RHSUSES -T21
EQMEDIUM
  LHSUSES T24
  RHSUSES -T24
EQNEAR
  LHSUSES T37
  RHSUSES -T37
EQNOT
  LHSUSES -T1 T2 T6
  RHSUSES -T6
EQON
  LHSUSES T34
  RHSUSES -T34
EQRED
  LHSUSES T7
  RHSUSES -T7
EQSMALL
  LHSUSES T27
  RHSUSES -T27
EQTABLE
  LHSUSES T47
  RHSUSES -T47
EQTHAT
  LHSUSES T63
  RHSUSES -T63
EQTHE
  LHSUSES G1 G2
  RHSUSES -G1 -G2
EQTHERE
  LHSUSES G9 G10 G17 -G18
  RHSUSES -G9 -G10
EQUNDER
  LHSUSES T39
  RHSUSES -T39
EQWHAT
  LHSUSES T57
  RHSUSES -T57
EQWHERE
  LHSUSES G21
  RHSUSES -G21
EQWHICH
  LHSUSES T60
  RHSUSES -T60
ERROR
  LHSUSES E2
  RHSUSES S7 -E2 A25 R5 P9 N10 N29 F2 F6 F51 F53 B17 B27 B57 M31 M53 V60
ERRORS
  LHSUSES E4 E6 E8
  RHSUSES E2 E4 -E4 -E6 E8 -E8
ERRREF
  LHSUSES E8 B1 B3 B57
  NESTEDL N31 N33
  RHSUSES -E8 G13 N31 N33 N41 N42 N43 N44 N45
FINDAMBIGP
  LHSUSES B43 B44 B45 B46
  RHSUSES B41 -B43 -B44 B45 -B45 -B46
FINDAMBIGR
  LHSUSES B33 B34 B35 B36
  RHSUSES B31 -B33 -B34 B35 -B35 -B36
FINDPOSS
  LHSUSES F13 F15 F21 F27 F31 F35 B13 B17 B23 B27 B31 B33 B34 B36 B41 B43 B44
  B46 B57 V14
  NESTEDL F11 F13 F15 B17 B27 B39 B36 B45 B46 B59 M1 M2 M6 M12 M16
  RHSUSES F1 F5 -F13 -F21 -F27 -F31 -F35
GBD
  LHSUSES N15 M1 M2 M5 -M11 -M12 -M51 -M60 V2
  RHSUSES G2 G7
GBE
  LHSUSES -M11 -M12 V20
  RHSUSES G9
GBQD
  LHSUSES A14 -M51 -M53
  NESTEDL A25
  RHSUSES G18
GBQE
  LHSUSES G5 -G6 G10 N8A -F53 M15 M16 -M51 -M53 V29
  NESTEDL N10
  RHSUSES G17
GBQW
  LHSUSES N16 -F53 B14 B24 -M11 -M12 V10 V12 V14
  NESTEDL B17 B27 B55 B57
  RHSUSES G13
GBQWE
  LHSUSES -M11 -M12 V17 V19
  RHSUSES G21
GTYPED
  LHSUSES G1 -G2 G6 -G7 -G9 -G13 -G17 -G18 -G21
  RHSUSES G2 G7 G9 G13 G17 G18 G21
HASAV
  LHSUSES E11 -F27 -F29 -F35 -B21 B23 B25 -B28 B29 B41 B43 B46 -B46 V35 V36 D11
   D12
  NESTEDL B21 B27 B46 M16 V37 D2 D3 D4 D11 D12
  RHSUSES N51 B21 M2 M5
HASREL
  LHSUSES E12 -F31 -B11 B13 B15 -B18 B19 B31 B33 B34 -B36 V17 V18 V30 V31
  NESTEDL B11 B17 B30 M12 V19 V32
  RHSUSES B11 M1
HASRELN
  LHSUSES B1 B3
  RHSUSES R11 R12
INDEFDET
  LHSUSES N5 N6 N23 -N29
  RHSUSES G6 G7
ISA
  LHSUSES E13 F1 F5 -F21 -F23 D4
  NESTEDL F2 F8
  RHSUSES N41 N42 N43 N44 N45
ISAV
  LHSUSES A1 A5 A15 N21 F41
  NESTEDL A25 N29
  RHSUSES A15 A17 A19
ISAVW
  LHSUSES A14 A15 A17 A19 A25
  RHSUSES T7 T10 T13 T16 T21 T24 T27 -A14 -A15 -A17 -A19
ISCOP
  LHSUSES G10 G17 G18 G31 G32 A17 R1 N8C N15 N16
  NESTEDL A25 R5 N10
  RHSUSES T1 T6
ISDEF
  LHSUSES A1 N8A N33
  NESTEDL N10
  RHSUSES N1 N2
ISINDEF
  LHSUSES A5 N31
  RHSUSES N5 N6 -N31
ISNOUN
  LHSUSES A14 R2 P1 N31 N33
  NESTEDL A25 R5 P9
  RHSUSES G13 N21 N22 N23
ISNOUNW
  LHSUSES G13 N21 N22 N23 N29
  RHSUSES T41 T44 T47 T50 T53 T57 -G13 -N21 -N22 -N23
ISPRED
  LHSUSES -A15 R3 -R5 P2 -P9 F41
  NESTEDL -A25
  RHSUSES A17
ISREL
  LHSUSES R11 R12 W50
  NESTEDL N10
  RHSUSES R1 R2 R3
 ISRELPRON
  LHSUSES P9 -N15 -N16
  RHSUSES P1 P2
 ISRELPRONW
  LHSUSES P1 P2
  RHSUSES T60 T63 -P1 -P2
 ISRELW
  LHSUSES R1 R2 R3 R5
  RHSUSES T31 T34 T37 T39 -R1 -R2 -R3
 LEFTOF
  LHSUSES S0 S1 S4 S7 T1 T2 T4 T57 E4 E6 G5 G10 G17 G18 A14 A15 A17 A18 A25 R1
  R2 R3 R5 P1 P2 P9 N8A N8B N8C N8D N10 N15 N16 N21 N22 N23 N29
  NESTEDL A25
  RHSUSES T4 -T4
 MAKISA
  LHSUSES N41 N42 N43 N44 N45
  RHSUSES N31 -N41 -N42 -N43 -N44 -N45
 NEWAV
  LHSUSES N51
  RHSUSES A5 -N51
 NEWOBJ
  LHSUSES -F5 B11 -B18 B21 -B28
  RHSUSES N41 N42 N43 N44 N45
 NPBOUND
  LHSUSES B51 B53 B55 B57
  RHSUSES S4 N15 N16 -B50
 NPBOUNDL
  LHSUSES B50
  RHSUSES N15 N16 -B50
 NPOCHK
  LHSUSES N8A N8B N8C N8D N10
  RHSUSES N1 N2 N5 N6 -N8A -N8B -N8C -N8D -N10
 NRESTR
  LHSUSES F21 F23
  RHSUSES N33 -F21 -F23
 NULLREF
  LHSUSES F51 F53 V12 V29
  RHSUSES F11 F23 F29
 OCHK
  LHSUSES F11 F13 F15
  RHSUSES -F11 -F13 -F15 F21 F27 F31 F35
 OLDAV
  LHSUSES -A1 -A5 -F41
  RHSUSES A1 A5 F41
 OLDREF
  LHSUSES -B1 -B3
  RHSUSES B1 B3
 OLDREL
  LHSUSES -R11 -R12
  RHSUSES R11 R12
 PREDINCON
  LHSUSES B48 M2 M15 M53
  RHSUSES E21 -B48 -M2 -M15
 PREDINCONT
  LHSUSES E21
  RHSUSES B28 B29
 PREDREDUN
  LHSUSES B41 M5 M16
  RHSUSES E22 -B43 -B44 -M5 -M16
 PREDREDUNT
  LHSUSES E22
  RHSUSES B25
 PREDRESTR
  LHSUSES F35
  RHSUSES E23
 PREDRESTRT
  LHSUSES E23
  RHSUSES B23 B24 B43 B44
 PREDRESTRCHK
  LHSUSES B21 B23 B24 B25 B27 B28 B29
  RHSUSES F41 -B21 -B23 -B24 -B25 -B27 -B28 -B29 B48
 QNOUN
  LHSUSES G13
  RHSUSES T57 -G13
 QWFIND
  LHSUSES F1 F2
  RHSUSES G13 -F1 -F2
 QWRDESCR2
  LHSUSES V18
  RHSUSES V17 -V18 V10
 QWREPLY
  LHSUSES V15
  RHSUSES V10 V14 -V15
 QWRPHRASE1
  LHSUSES D22 D23 D24
  NESTEDL D21
  RHSUSES D21 D22 -D22 D23 -D23 -D24
 QWRPHRASE2
  LHSUSES D26 D27 D28
  NESTEDL D25
  RHSUSES D25 D26 -D26 D27 -D27 -D28
 QWRREPLY1
  LHSUSES D21 D22 D23
  NESTEDL D21 D22 D23 D24
  RHSUSES V17 -D22 -D23
 QWRREPLY2
  LHSUSES D25 D26 D27
  NESTEDL D26 D27 D28
  RHSUSES V18 -D26 -D27
 QWRREPLY3
  LHSUSES D29
  RHSUSES V19 -D29
 REFERS
  LHSUSES F23 F29 B1 B3 B15 B18 B19 B25 B28 B29 B33 B34 B51 B53 M1 M2 M5 V10
  V17 V19 V30 V31 V32 V35 V36 V37
  RHSUSES N41 N42 N43 N44 N45 F13 -F23 -F29
 RELINCON
  LHSUSES B38 B39 M1 M11 M51
  RHSUSES E31 -B38 -B39 -M1 -M11
 RELINCONT
  LHSUSES E31
  RHSUSES B18 B19
 RELREDUN
  LHSUSES B31 M12
  RHSUSES E32 -B33 -B34 -M12
 RELREDUNT
  LHSUSES E32
  RHSUSES B15
 RELRESTR
  LHSUSES F31
  RHSUSES E33
 RELRESTRT
  LHSUSES E33
  RHSUSES B13 B14 B33 B34
 RELRESTRCHK
  LHSUSES B11 B13 B14 B15 B17 B18 B19
  RHSUSES B1 B3 -B11 -B13 -B14 -B15 -B17 -B18 -B19 B38 B39
 REPLY
  LHSUSES V5
  RHSUSES E2 E6 V2 V12 V15 V20 V25 V30 V31 V32 V35 V36 V37 D24 D28 D29
 SCAN
  LHSUSES -S1 -S4 T1 T2 T4 T7 T10 T13 T16 T21 T24 T27 T31 T34 T37 T39 T41 T44
  T47 T50 T53 T57 T60 T63 G1 G2 G5 G6 G7 G9 G10 G21
  NESTEDL -S7
  RHSUSES S0 S1 -S7 -T1 -T2 -T4 -T7 -T10 -T13 -T16 -T21 -T24 -T27 -T31 -T34
  -T37 -T39 -T41 -T44 -T47 -T50 -T53 -T57 -T60 -T63 -G1 -G2 -G5 -G6 -G7 -G9
  -G10 -G21
 SCANFIN
  LHSUSES S0 S1 S4 S7 V5
  RHSUSES S0 -S0 S1 -S1 -S4 -S7 -V5
 SENTBOUND
  LHSUSES V2 V10 V12 V14 V17 V19 V20 V30 V31 V32 V35 V36 V37 V40 V42 V44 V46
  V48 V49
  RHSUSES S4
 SENTENCE
  LHSUSES S4 G1 G2 G5 G6 G7 G9 G13 G17 G18 G21 N15 N16 F53 B17 B27 M11 M12 M51
  M53
 TEST1B
 TEXT
  LHSUSES S0
 TRACING
  RHSUSES S0 E11 E12 E13 E21 E22 E23 E31 E32 E33 F13 F15
 WORDEQ
  LHSUSES E4 E6 G5 N8A
  NESTEDL N10
  RHSUSES T1 T4 T7 T10 T13 T16 T21 T24 T27 T31 T34 T37 T39 T41 T44 T47 T50 T53
  T57 T60 T63 G1 G2 G5 G6 G7 G9 G10 G21
```

Appendix C. [illegible]

```
FINAL RUN WITH PROGRAM TRACE

1 INPUT TEXT IS " A LARGE GREEN BLOCK IS ON A RED TABLE "
ADDING SIZE LAPGE (POS) TO BLOCK-1
ADDING COLOR GPEEN (POS) TO BLOCK-1
ADDING BLOCK BLOCK-1
ADDING COLOR RED (POS) TO TABLE-1
ADDING TABLE TABLE-1
ADDING BLOCK-1 ON TABLE-1 (POS)
REPLY ((OKAY))

ISA (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (TABLE-1 COLOR RED POS)
HASREL (BLOCK-1 ON TABLE-1 POS)

2 INPUT TEXT IS " A BLUE BALL IS ON THE TABLE "
ADDING COLOR BLUE (POS) TO BALL-1
ADDING BALL BALL-1
OBJ-2 REFERS TABLE-1
ADDING BALL-1 ON TABLE-1 (POS)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOP BLUE POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (TABLE-1 COLOP RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BLOCK-1 ON TABLE-1 POS)

3 INPUT TEXT IS " THE BALL IS NEAR THE BLOCK "
OBJ-1 REFERS BALL-1
OBJ-2 REFERS BLOCK-1
RELINCON OBJ-1 B2-1 NEAR BLOCK-1 POS
ADDING BALL-1 NEAR BLOCK-1 (POS)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (TABLE-1 COLOR PED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS)

4 INPUT TEXT IS " A BLUE BALL IS ON THE BLOCK "
ADDING COLOP BLUE (POS) TO BALL-2
ADDING BALL BALL-2
OBJ-2 REFERS BLOCK-1
ADDING BALL-2 ON BLOCK-1 (POS)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BLOCK-1 SIZE LARGE POS)
  (BLOCK-1 COLOP GPEEN POS) (TABLE-1 COLOR PED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS)

5 INPUT TEXT IS " THE BALL ON THE BLOCK IS SMALL "
OBJ-1 AMBIG B2-1 BALL-1 BALL-2 ...
OBJ-2 PEFEPS BLOCK-1
RELRESTR OBJ-1 B2-1 ON BLOCK-1 POS
OBJ-1 PEFERS BALL-2
PREDINCON OBJ-1 S7-1 SIZE SMALL POS
ADDING SIZE SMALL (POS) TO BALL-2
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOP BLUE POS) (BALL-2 COLOP BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LAPGE POS) (BLOCK-1 COLOP GPEEN POS) (TABLE-1 COLOP RED POS)
HASPEL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS)

2

RUN TIME 1 MIN. 19.8 SEC

EXAM      TRY      FIPE      UNACT   E/F      E/T      T/F
2412      498      259       802     9.31     4.86     1.92
0.0331    0.161    0.308     0.0994  SEC AVG

537 INSERTS 285 DELETES 59 WARNINGS 7 NEW OBJECTS
MAX :SMPX LENGTH 103
CORE (FREE.FULL): (8588 . 1593) USED (2187 . 273)

:ACTS LOADPS (MILIPS . EXP) (MILXI . EXP) (MILIM . MAC) (MILGARP . EXP) (MILN
  . EXP) (MILFB . EXP) (MILM . EXP) (MILVB . EXP) MILC RUN SMPXEMPTY SMPXEMPTY
  SMPXEMPTY SMPXEMPTY SMPXEMPTY

TRACE
(X1-1
S0-1 G7-1 N6-1 N90-1
S1-1 T21-1 A19-1 A6-1
S1-2 T10-1 A15-1 A5-2
S1-3 T44-1 N21-1 N31-1 N42-1 N61-1 N61-2 E11-1 E11-2 E13-1
S1-4 T1-1 N15-1 B59-1 G32-1
S1-5 T34-1 R1-1 R11-1
S1-6 G5-1 N5-1 N98-1
S1-7 T7-1 A19-2 A5-3
S1-8 T47-1 N21-2 N31-2 N43-1 N61-3 E11-3 E13-2 B1-1 B11-1 E12-1
S4-1 B53-1 B51-1 B56-1 V2-1 X2-1
S0-2 G7-2 N6-2 N90-2
S1-9 T13-1 A19-3 A6-4
S1-10 T41-1 N21-3 N31-3 N41-1 N61-4 E11-4 E13-3
S1-11 T1-2 N15-2 B59-2 G32-2
S1-12 T34-2 R1-2 R11-2
S1-13 G1-1 N1-1 N98-2 F5-1 F5-2
S1-14 T47-2 N22-1 N33-1 F21-1 F13-1 B1-2 B11-2 E12-2
S4-2 B51-2 B53-2 B55-2 V2-2 X3-1
S0-3 G2-1 N2-1 N90-3 F5-3 F5-4 F5-5
S1-15 T41-2 N22-2 N33-2 F21-2 F21-3 F13-2
S1-16 T1-3 N15-3 B55-3 B59-3 G32-3
S1-17 T37-1 R1-3 R11-3
S1-18 G1-2 N1-2 N98-3 F5-6 F5-7 F5-8
S1-19 T44-2 N22-3 N33-3 F21-4 F21-5 F13-3 B1-3 B10-1 E31-1 M1-1 E12-3
S4-3 B53-3 B51-3 B55-4 V2-3 X4-1
S0-4 G7-3 N6-3 N90-4
S1-20 T13-2 A19-4 A5-5
S1-21 T41-3 N21-4 N31-4 N41-2 N51-5 E11-5 E13-4
S1-22 T1-4 G32-4 N15-4 B59-4
S1-23 T34-3 R1-4 R11-4
S1-24 G1-3 N1-3 N98-4 F5-9 F5-10 F5-11
S1-25 T44-3 N22-4 N33-4 F21-6 F21-7 F13-4 B1-4 B11-3 E12-4
S4-4 B51-4 B53-4 B55-5 V2-4 X5-1
S0-5 G2-2 N2-2 N90-5 F5-12 F5-13 F5-14 F5-15
S1-26 T41-4 N22-5 N33-5 F21-8 F21-9 F15-1
S1-27 T34-4 R2-1 R12-1
S1-28 G1-4 N1-4 N98-5 F5-16 F5-17 F5-18 F5-19
S1-29 T44-4 N22-6 N33-6 F21-10 F21-11 F21-12 F13-5 B1-5 B13-1 E30-1 F31-1 F13-6
S1-30 T1-5 N15-5 B51-5 B53-5 B55-6 B59-5 G32-5
S1-31 T27-1 A17-1 F41-1 B28-1 E21-1 M2-1 E11-6
S4-5 B55-7 V2-5)

FIRED 71 OUT OF 193 PRODS

                    - - - - - - - - - - - - - - - - - - - -

SECOND SEGMENT

6 INPUT TEXT IS " WHAT IS ON THE BLOCK "
OBJ-2 REFEPS BLOCK-1
RELRESTP OBJ-1 W1-1 ON BLOCK-1 POS
OBJ-1 REFERS BALL-2
REPLY ((THE SMALL BLUE BALL))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LAPGE POS) (BLOCK-1 COLOR GPEEN POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS
  (BLOCK-1 ON TABLE-1 POS)

7 INPUT TEXT IS " WHAT IS BLUE "
PREDRESTR OBJ-1 B3-1 COLOR BLUE POS
OBJ-1 AMBIG B3-1 BALL-1 BALL-2 ...
REPLY ((THE BLUE BALL)) ((THE SMALL BLUE BALL))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOP GREEN POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS
  (BLOCK-1 ON TABLE-1 POS)

8 INPUT TEXT IS " THERE IS A BOX ON THE TABLE "
ADDING BOY BOX-1
OBJ-2 PEFEPS TABLE-1
ADDING BOX-1 ON TABLE-1 (POS)
REPLY ((OKAY))
```

```
ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS)

9 INPUT TEXT IS " IS THE BOX ON THE TABLE "
OBJ-1 REFERS BOX-1
OBJ-2 REFERS TABLE-1
RELREDUN OBJ-1 B3-1 ON TABLE-1 POS
REPLY ((YES))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (TABLE-1 TABLE)
HASAV (BALL-1 COLOP BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GPEEN POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS)

10 INPUT TEXT IS " IS THE BOX ON THE TABLE NEAR THE BLOCK "
OBJ-1 REFERS BOX-1
OBJ-2 REFERS TABLE-1
RELPEDUN OBJ-1 B3-1 ON TABLE-1 POS
OBJ-3 PEFEPS BLOCK-1
RELINCON OBJ-2 T6-1 NEAR BLOCK-1 POS
RELINCON OBJ-1 T6-1 NEAR BLOCK-1 POS
REPLY ((NO INFOPMATION ON RELATION NEAR))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOP BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOP GPEEN POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS)

2

RUN TIME 1 MIN. 21.5 SEC

EXAM     TRY     FIRE     WMACT   E/F     E/T     T/F
2320     400     200      043     0.29    4.96    1.67
0.0351   0.174   0.291    0.0967  SEC AVG

530 INSERTS 305 DELETES 106 WARNINGS 9 NEW OBJECTS
MAX :SMPX LENGTH 102
CORE (FREE.FULL): (6151 . 1539) USED (2130 . 303)

:ACTS SAVEDB (CLOSED (MIL13 . DBS)) (CLOSED (MIL13 . TRS)) LOADPS (MILX2 . EXP)
   RUN SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY

TRACE
(X0-1
S0-1 T57-1 G13-1 F1-1 F1-2 F1-3 F1-4
S1-1 T1-1 G32-1 N16-1 B55-1 B59-1
S1-2 T34-1 R1-1 R11-1
S1-3 G1-1 N1-1 N90-1 F5-1 F5-2 F5-3 F5-4
S1-4 T44-1 N22-1 N33-1 F21-1 F21-2 F21-3 F13-1 B1-1 B13-1 E33-1 F31-1 F31-2
  F31-3 F13-2
S4-1 B53-1 B51-1 B55-2 V10-1 D1-1 D11-1 D2-1 D3-1 D11-2 D2-2 D4-1 V15-1 X7-1
S0-2 T57-2 G13-2 F1-5 F1-6 F1-7 F1-8
S1-5 T1-2 N16-2 B55-3 B59-2 G32-2
S1-6 T13-1 A17-1 F41-1 B23-1 B23-2 E23-1 F35-1 F35-2 F15-1
S4-2 B55-4 V14-1 V14-2 D1-2 D1-3 D2-3 D11-3 D2-4 D3-2 D3-3 D11-4 D11-5 D2-5 D2-6
  D4-2 D4-3 V15-2 V15-3 X8-1
S0-3 G9-1
S1-7 T1-3 G32-3
S1-8 G6-1 N6-1 N9C-1
S1-9 T53-1 N23-1 N31-1 N45-1 E13-1
S1-10 T34-2 R2-1 R11-2
S1-11 G1-2 N1-2 N9A-2 F5-5 F5-6 F5-7 F5-8
S1-12 T47-1 N22-2 N33-2 F21-4 F21-5 F21-6 F13-3 B1-2 B11-1 E12-1
S4-3 B51-2 B53-2 B55-5 V29-1 X9-1
S0-4 T1-4 G10-1 G32-4
S1-13 G1-3 N2-1 N9C-2 F5-9 F5-10 F5-11 F5-12 F5-13
S1-14 T53-2 N22-3 N33-3 F21-7 F21-8 F21-9 F21-10 F13-4
S1-15 T34-3 R2-2 R11-3
S1-16 G1-4 N1-3 N9A-3 F5-14 F5-15 F5-16 F5-17 F5-18
S1-17 T47-2 N22-4 N33-4 F21-11 F21-12 F21-13 F21-14 F13-5 B1-3 B15-1 E32-1 M12-1
S4-4 B51-3 B53-3 B55-6 V46-1 V20-1 X10-1
S0-5 T1-5 G10-2 G32-5
S1-18 G1-5 N2-2 N9C-3 F5-19 F5-20 F5-21 F5-22 F5-23
S1-19 T53-3 N22-5 N33-5 F21-15 F21-16 F21-17 F21-18 F13-6
S1-20 T34-4 R2-3 R11-4
S1-21 G1-6 N1-4 N9A-4 F5-24 F5-25 F5-26 F5-27 F5-28
S1-22 T47-3 N22-6 N33-6 F21-19 F21-20 F21-21 F21-22 F13-7 B1-4 B15-2 E32-2 M12-2
S1-23 T37-1 R2-4 R12-1
S1-24 G1-7 N1-5 N90-5 F5-29 F5-30 F5-31 F5-32 F5-33
S1-25 T44-2 N22-7 N33-7 F21-23 F21-24 F21-25 F21-26 F13-8 B1-5 B10-1 E31-1 B
  B10-2 E31-2 M11-1
S4-5 B51-4 B53-4 B55-7 V42-1 V32-1)

FIRED 70 OUT OF 193 PRODS

                    - - - - - - - - - - - - - - - - - - - -

THIRD SEGMENT

11 INPUT TEXT IS " THERE IS A BOX ON A RED FLOOR WHICH IS NOT RED "
ADDING BOX BOX-2
ADDING COLOR RED (POS) TO FLOOR-1
ADDING FLOOR FLOOR-1
ADDING BOX-2 ON FLOOR-1 (POS)
PREDINCON FLOOR-1 R12-1 COLOR RED NEG
ADDING COLOR RED (NEG) TO BOX-2
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX)
  (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG)
  (FLOOR-1 COLOP RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON [illegible]
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)

12 INPUT TEXT IS " WHAT IS NOT BLUE "
PREDRESTR OBJ-1 B4-1 COLOR BLUE NEG
REPLY ((NOTHING))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX)
  (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG)
  (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)

13 INPUT TEXT IS " THE BOX THAT IS NOT RED IS NOT ON THE TABLE "
OBJ-1 AMBIG B2-1 BOX-1 BOX-2 ...
PREDPESTR OBJ-1 R6-1 COLOR RED NEG
OBJ-1 REFERS BOX-2
OBJ-2 REFERS TABLE-1
RELINCON OBJ-1 B2-1 ON TABLE-1 NEG
ADDING BOX-2 ON TABLE-1 (NEG)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX)
  (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE [illegible]
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG)
  (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

14 INPUT TEXT IS " WHAT IS NOT ON THE TABLE "
OBJ-2 REFERS TABLE-1
RELRESTR OBJ-1 W1-1 ON TABLE-1 NEG
OBJ-1 REFERS BOX-2
REPLY ((THE UN- RED BOX))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX)
  (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOP BLUE POS) (BALL-2 COLOP BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LAPGE POS) (BLOCK-1 COLOR GPEEN POS) (BOX-2 COLOR RED NEG)
  (FLOOR-1 COLOP RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

15 INPUT TEXT IS " IS THERE A BLACK BOX ON THE FLOOR "
REPLY ((NO))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX)
  (FLOOR-1 FLOOP) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG)
  (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
MAGREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

2

RUN TIME 1 MIN. 20.9 SEC

EXAM     TRY     FIRE     WMACT   E/F     E/T     T/F
2081     406     276      864     7.47    4.43    1.80
0.0382  0.174   0.290    0.0847  SEC AVG

545 INSERTS 309 DELETES 112 WARNINGS 12 NEW OBJECTS
MAX :SMPX LENGTH 102
CORE (FREE.FULL): (5037 . 1524) USED (2150 . 280)

(ACTS SAVEDB (CLOSED (MIL24 . DBS)) (CLOSED (MIL25 . TRS)) LOADPS (MILX3 . EXP)
   RUN SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY

TRACE
(X11-1
S0-1 G9-1
S1-1 T1-1 G32-1
S1-2 G6-1 N6-1 N9C-1
S1-3 T53-1 N23-1 N31-1 N45-1 E13-1
S1-4 T34-1 R2-1 R11-1
S1-5 G6-2 N5-1 N90-1
S1-6 T7-1 A19-1 A5-1
S1-7 T50-1 N21-1 N31-2 N44-1 N61-1 E11-1 E13-2 B1-1 B11-1 E12-1
S1-8 T60-1 P1-1
S1-9 T2-1
S1-10 T4-1 G31-1
S1-11 T7-2 A17-1 F41-1 B29-1 E21-1 B40-1 B21-1 E11-2
S4-1 B55-1 V20-1 X12-1
S0-2 T57-1 G13-1 F1-1 F1-2 F1-3 F1-4 F1-5 F1-6 F1-7
S1-12 T2-2
S1-13 T4-2 N16-1 B55-2 B59-1 G31-2
S1-14 T13-1 A17-2 F41-2 B24-1 E23-1 F35-1 F35-2 F35-3 F35-4 F35-5 F35-6 F35-7
  F11-1
S4-2 B55-3 V12-1 X13-1
S0-3 G2-1 N2-1 N9D-1 F5-1 F5-2 F5-3 F5-4 F5-5 F5-6 F5-7
S1-15 T53-2 N22-1 N33-1 F21-1 F21-2 F21-3 F21-4 F21-5 F15-1
S1-16 T63-1 P1-2
S1-17 T2-3
S1-18 T4-3 G31-3
S1-19 T7-3 A17-3 F41-3 B23-1 E23-2 F35-8 F13-1
S1-20 T2-4
S1-21 T4-4 G31-4 N15-1 B55-4 B59-2
S1-22 T34-2 R1-1 R11-2
S1-23 G1-1 N1-1 N58-2 F5-8 F5-9 F5-10 F5-11 F5-12 F5-13 F5-14
S1-24 T47-1 N22-2 N33-2 F21-6 F21-7 F21-8 F21-9 F21-10 F21-11 F13-2 B1-2 B18-1
  E31-1 M1-1 E12-2
S4-3 B53-1 B51-1 B55-5 V2-1 X14-1
S0-4 T57-2 G13-2 F1-8 F1-9 F1-10 F1-11 F1-12 F1-13 F1-14
S1-25 T2-5
S1-26 T4-5 G31-5 N16-2 B55-6 B59-3
S1-27 T34-3 R1-2 R11-3
S1-28 G1-2 N1-2 N90-3 F5-15 F5-16 F5-17 F5-18 F5-19 F5-20 F5-21
S1-29 T47-2 N22-3 N33-3 F21-12 F21-13 F21-14 F21-15 F21-16 F21-17 F13-3 B1-3
  B18-1 E33-1 F31-1 F31-2 F31-3 F31-4 F31-5 F31-6 F13-4
S4-4 B53-2 B51-2 B55-7 V10-1 D1-1 D2-1 D3-1 D2-2 D12-1 D4-1 V15-1 X15-1
S0-5 T1-2 G17-1 G32-2
S1-30 G18-1
S1-31 G5-1 N2-2 N9A-1 F5-22 F5-23 F5-24 F5-25 F5-26 F5-27 F5-28
S1-32 T18-1 A19-2 A1-1 F27-1 F27-2 F27-3 F27-4 F27-5 F27-6 F27-7 F11-2 V25-1
  V5-1)

FIRED 100 OUT OF 193 PRODS

                    - - - - - - - - - - - - - - - - - - - -

(THIRD SEGMENT)

          S T E G A R P N F B M V D X

X11-1                               X   1.
S0-1      S                             1.
G9-1            G                       1.
S1-1      S                             1.
T1-1        T                           1.
G32-1           G                       1.
S1-2      S                             1.
G6-1            G                       1.
N6-1                    N               2..
S1-3      S                             1.
T53-1       T                           1.
N23-1                   N               3...
E13-1         E                         1.
S1-4      S                             1.
T34-1       T                           1.
R2-1                R                   2..
S1-5      S                             1.
G6-2            G                       1.
N5-1                    N               2..
S1-6      S                             1.
T7-1        T                           1.
A19-1             A                     2..
S1-7      S                             1.
T50-1       T                           1.
N21-1                   N               4....
E11-1         E                         2..
B1-1                        B           2..
E12-1         E                         1.
S1-8      S                             1.
T60-1       T                           1.
P1-1                  P                 1.
S1-9      S                             1.
T2-1        T                           1.
S1-10     S                             1.
T4-1        T                           1.
G31-1           G                       1.
S1-11     S                             1.
T7-2        T                           1.
A17-1             A                     1.
F41-1                     F             1.
B29-1                       B           1.
E21-1         E                         1.
B40-1                       B           2..
E11-2         E                         1.
S4-1      S                             1.
B55-1                       B           1.
V20-1                           V       1.
X12-1                               X   1.
S0-2      S                             1.
T57-1       T                           1.
G13-1           G                       1.
F1-1                      F             7.......
S1-12     S                             1.
T2-2        T                           1.
S1-13     S                             1.
T4-2        T                           1.
N16-1                   N               1.
B55-2                       B           2..
G31-2           G                       1.
S1-14     S                             1.
T13-1       T                           1.
A17-2             A                     1.
F41-2                     F             1.
B24-1                       B           1.
E23-1         E                         1.
F35-1                     F             8........
S4-2      S                             1.
B55-3                       B           1.
V12-1                           V       1.
X13-1                               X   1.
S0-3      S                             1.
G2-1            G                       1.
N2-1                    N               2..
F5-1                      F             7.......
S1-15     S                             1.
T53-2       T                           1.
N22-1                   N               2..
F21-1                     F             5.....
S1-16     S                             1.
T63-1       T                           1.
P1-2                  P                 1.
S1-17     S                             1.
T2-3        T                           1.
S1-18     S                             1.
T4-3        T                           1.
G31-3           G                       1.
S1-19     S                             1.
T7-3        T                           1.
```

```
A17-3           A                 1.
F41-3                  F          1.
B23-1                    B        1.
E23-2       E                     1.
F36-8                  F          2..
S1-20    S                        1.
T2-4       T                      1.
S1-21    S                        1.
T4-4       T                      1.
G31-4         G                   1.
N15-1                N            1.
B55-4                    B        2..
S1-22    S                        1.
T34-2      T                      1.
R1-1              R               2..
S1-23    S                        1.
G1-1          G                   1.
N1-1                 N            2..
F5-8                   F          7.......
S1-24    S                        1.
T47-1      T                      1.
N22-2                N            2..
F21-6                  F          7.......
B1-2                     B        2..
E31-1       E                     1.
M1-1                       M      1.
E12-2       E                     1.
S4-3     S                        1.
B53-1                    B        3...
V2-1                        V     1.
X14-1                          X  1.
S0-4     S                        1.
T57-2      T                      1.
G13-2         G                   1.
F1-8                   F          7.......
S1-25    S                        1.
T2-5       T                      1.
S1-26    S                        1.
T4-5       T                      1.
G31-5         G                   1.
N16-2                N            1.
B55-6                    B        2..
S1-27    S                        1.
T34-3      T                      1.
R1-2              R               2..
S1-28    S                        1.
G1-2          G                   1.
N1-2                 N            2..
F5-15                  F          7.......
S1-29    S                        1.
T47-2      T                      1.
N22-3                N            2..
F21-12                 F          7.......
B1-3                     B        2..
E33-1       E                     1.
F31-1                  F          7.......
S4-4     S                        1.
B53-2                    B        3...
V10-1                       V     1.
D1-1                          D   6......
V15-1                       V     1.
X15-1                          X  1.
S0-5     S                        1.
T1-2       T                      1.
G17-1         G                   2..
S1-30    S                        1.
G18-1         G                   1.
S1-31    S                        1.
G5-1          G                   1.
N2-2                 N            2..
F5-22                  F          7.......
S1-32    S                        1.
T16-1      T                      1.
A19-2           A                 2..
F27-1                  F          8........
V25-1                       V     2..

PERCENTAGES OF FIRINGS OF EACH TYPE, OUT OF TOTAL 278
S 14..............
T 10..........
E 3...
G 6......
A 2..
R 2..
P 0
N 10..........
F 32................................
B 9.........
M 0
V 2..
D 2..
X 1.
```

- - - - - - - - - - - - - - - - - - - -

```
FOURTH SEGMENT

16 INPUT TEXT IS " WHAT THAT IS NOT RED IS ON THE FLOOR "
PREDRESTR OBJ-1 RS-1 COLOR RED NEG
OBJ-1 REFERS BOX-2
OBJ-2 REFERS FLOOR-1
RELREDUN OBJ-1 H1-1 ON FLOOR-1 POS
REPLY ((THE UN- RED BOX))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX)
  (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG)
  (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

17 INPUT TEXT IS " WHAT IS "
REPLY ((THE BLUE BALL)) ((THE BOX)) ((THE UN- RED BOX)) ((THE RED FLOOR))
  ((THE RED TABLE)) ((THE SMALL BLUE BALL)) ((THE LARGE GREEN BLOCK))

ISA (BALL-1 BALL) (BALL-2 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX)
  (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG)
  (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

18 INPUT TEXT IS " A SMALL RED BALL IS IN THE BOX ON THE RED FLOOR "
ADDING SIZE SMALL (POS) TO BALL-3
ADDING COLOR RED (POS) TO BALL-3
ADDING BALL BALL-3
OBJ-2 AMBIG B8-1 BOX-1 BOX-2 ...
OBJ-3 AMBIG R11-1 FLOOR-1 TABLE-1 ...
OBJ-3 REFERS FLOOR-1
RELRESTR OBJ-2 B8-1 ON FLOOR-1 POS
OBJ-2 REFERS BOX-2
ADDING BALL-3 IN BOX-2 (POS)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BLOCK-1 BLOCK) (BOX-1 BOX)
  (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BLOCK-1 SIZE LARGE POS)
  (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS)
  (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS)
  (BOX-2 ON FLOOR-1 POS) (BOX-2 ON TABLE-1 NEG)

19 INPUT TEXT IS " THERE IS A LARGE GREEN BALL IN THE BOX ON THE FLOOR NEAR THE
  BALL IN THE BOX ON THE FLOOR "
ADDING SIZE LARGE (POS) TO BALL-4
ADDING COLOR GREEN (POS) TO BALL-4
ADDING BALL BALL-4
OBJ-2 AMBIG B9-1 BOX-1 BOX-2 ...
OBJ-3 REFERS FLOOR-1
RELRESTR OBJ-2 B9-1 ON FLOOR-1 POS
OBJ-2 REFERS BOX-2
ADDING BALL-4 IN BOX-2 (POS)
OBJ-4 AMBIG B15-1 BALL-1 BALL-2 ...
OBJ-5 AMBIG B18-1 BOX-1 BOX-2 ...
OBJ-6 REFERS FLOOR-1
RELRESTR OBJ-5 B18-1 ON FLOOR-1 POS
OBJ-5 REFERS BOX-2
```

```
RELRESTR OBJ-4 B15-1 IN BOX-2 POS
OBJ-4 REFERS BALL-3
RELINCON OBJ-3 F12-1 NEAR BALL-3 POS
RELINCON OBJ-2 F12-1 NEAR BALL-3 POS
ADDING BALL-4 NEAR BALL-3 (POS)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BLOCK-1 BLOCK)
  (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-4 IN BOX-2 POS) (BALL-4 NEAR BALL-3 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

2

RUN TIME 2 MIN. 37.3 SEC

EXAM     TRY      FIRE     WMACT  E/F     E/T     T/F
2851     711      452      1306   6.31    4.01    1.57
0.0552  0.221    0.340    0.120   SEC AVG

802 INSERTS 504 DELETES 199 WARNINGS 14 NEW OBJECTS
MAX ;SMPX LENGTH 102
CORE (FREE.FULL): (4320 . 1300) USED (3492 . 492)

;ACTS SAVEDB (CLOSED (MIL33 . DBS)) (CLOSED (MIL33 . TRS)) LOADPS (MILX4 . EXP)
   RUN SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY

TRACE
(X18-1
S0-1 T57-1 G13-1 F1-1 F1-2 F1-3 F1-4 F1-5 F1-6 F1-7
S1-1 T63-1 P1-1
S1-2 T2-1
S1-3 T4-1 G31-1
S1-4 T7-1 A17-1 F41-1 B23-1 E23-1 F35-1 F35-2 F35-3 F35-4 F35-5 F35-6 F13-1
S1-5 T1-1 N18-1 B55-1 B59-1 G32-1
S1-6 T34-1 R1-1 R11-1
S1-7 G1-1 N1-1 N98-1 FS-1 FS-2 FS-3 FS-4 FS-5 FS-6 FS-7
S1-8 T50-1 N22-1 N33-1 F21-1 F21-2 F21-3 F21-4 F21-5 F21-6 F13-2 B1-1 B15-1
  E32-1
S4-1 B55-2 V10-1 D1-1 D2-1 D3-1 D2-2 D12-1 D4-1 V15-1 B51-1 B53-1 X17-1
S0-2 T57-2 G13-2 F1-8 F1-9 F1-10 F1-11 F1-12 F1-13 F1-14
S1-9 T1-2 G32-2 N16-2 B55-3 B59-2
S4-2 B55-4 V14-1 V14-2 V14-3 V14-4 V14-5 V14-6 V14-7 D1-2 D1-3 D1-4 D1-5 D1-6
  D1-7 D1-8 D2-3 D2-4 D2-5 D2-6 D2-7 D11-1 D11-2 D3-2 D3-3 D3-4 D3-5 D3-6 D11-3
  D11-4 D11-5 D2-8 D2-9 D2-10 D2-11 D2-12 D2-13 D2-14 D12-2 D4-2 D4-3 D4-4 D4-5
  D4-6 V15-2 V15-3 V15-4 V16-5 V15-6 D3-7 D3-8 D11-6 D11-7 D2-15 D2-16 D4-7 D4-8
  V15-7 V15-8 X18-1
S0-3 G7-1 N8-1 N90-1
S1-10 T27-1 A19-1 A5-1
S1-11 T7-2 A15-1 A5-2
S1-12 T41-1 N21-1 N31-1 N41-1 N51-1 N61-2 E11-1 E11-2 E13-1
S1-13 T1-3 G32-3 N15-1 B59-3
S1-14 T31-1 R1-2 R11-2
S1-15 G1-2 N1-2 N98-2 FS-8 FS-9 FS-10 FS-11 FS-12 FS-13 FS-14
S1-16 T53-1 N22-2 N33-2 F21-7 F21-8 F21-9 F21-10 F21-11 F15-1
S1-17 T34-2 R2-1 P12-1
S1-18 G1-3 N1-3 N98-3 FS-15 FS-16 FS-17 FS-18 FS-19 FS-20 FS-21
S1-19 T7-3 A19-2 A1-1 F27-1 F27-2 F27-3 F27-4 F27-5 F15-2
S1-20 T50-2 N21-2 N33-3 F21-12 F13-3 B1-2 B13-1 E33-1 F31-1 F13-4 B3-1 B11-1
  E12-1
S4-3 B53-2 B53-3 B51-2 B55-5 V2-1 X19-1
S0-4 G9-1
S1-21 T1-4 G32-4
S1-22 G6-1 N8-2 N9C-1
S1-23 T21-1 A19-3 A5-3
S1-24 T18-1 A15-2 A5-4
S1-25 T41-2 N21-3 N31-2 N41-2 N51-3 N61-4 E11-3 E11-4 E13-2
S1-26 T31-2 R2-2 P11-3
S1-27 G1-4 N1-4 N98-4 FS-22 FS-23 FS-24 FS-25 FS-26 FS-27 FS-28 FS-29
S1-28 T53-2 N22-3 N33-4 F21-13 F21-14 F21-15 F21-16 F21-17 F21-18 F15-3
S1-29 T34-3 R2-3 P12-2
S1-30 G1-5 N1-5 N98-5 FS-30 FS-31 FS-32 FS-33 FS-34 FS-35 FS-36 FS-37
S1-31 T50-3 N22-4 N33-5 F21-19 F21-20 F21-21 F21-22 F21-23 F21-24 F21-25 F13-5
  B1-3 B13-2 E33-2 F31-2 F13-6 B3-2 B11-2 E12-2
S1-32 T37-1 P2-4 P12-3
S1-33 G1-6 N1-6 N98-6 FS-38 FS-39 FS-40 FS-41 FS-42 FS-43 FS-44 FS-45
S1-34 T41-3 N22-5 N33-6 F21-26 F21-27 F21-28 F21-29 F21-30 F15-4
S1-35 T31-3 R2-5 R12-4
S1-36 G1-7 N1-7 N98-7 FS-46 FS-47 FS-48 FS-49 FS-50 FS-51 FS-52 FS-53
S1-37 T53-3 N22-6 N33-7 F21-31 F21-32 F21-33 F21-34 F21-35 F21-36 F15-5
S1-38 T34-4 R2-6 R12-5
S1-39 G1-8 N1-8 N98-8 FS-54 FS-55 FS-56 FS-57 FS-58 FS-59 FS-60 FS-61
S1-40 T50-4 N22-7 N33-8 F21-37 F21-38 F21-39 F21-40 F21-41 F21-42 F21-43 F13-7
  B1-4 B13-3 E33-3 F31-3 F13-8 B3-3 B13-4 E33-4 F31-4 F31-5 F13-9 B3-4 B10-1
  E31-1 B39-1 B10-2 E31-2 B39-2 B11-3 E12-3
S4-4 B53-4 B53-5 B53-6 B51-3 B55-6 V20-1)

FIRED 91 OUT OF 192 PRODS
```

- - - - - - - - - - - - - - - - - - - -

```
FIFTH SEGMENT

20 INPUT TEXT IS " WHERE IS THE BOX THAT IS NOT RED "
OBJ-1 AMBIG B4-1 BOX-1 BOX-2 ...
PREDRESTR OBJ-1 R8-1 COLOR RED NEG
OBJ-1 REFERS BOX-2
REPLY ((THE UN- RED BOX IS ON THE RED FLOOR AND NOT ON THE RED TABLE))
  ((THE SMALL RED BALL IS IN IT)) ((THE LARGE GREEN BALL IS IN IT))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BLOCK-1 BLOCK)
  (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-4 IN BOX-2 POS) (BALL-4 NEAR BALL-3 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

21 INPUT TEXT IS " WHERE IS THE BALL IN THE BOX ON THE RED FLOOR THAT IS LARGE "
OBJ-1 AMBIG B4-1 BALL-1 BALL-2 ...
OBJ-2 AMBIG B7-1 BOX-1 BOX-2 ...
OBJ-3 AMBIG R10-1 BALL-3 FLOOR-1 ...
OBJ-3 REFERS FLOOR-1
RELRESTR OBJ-2 B7-1 ON FLOOR-1 POS
OBJ-2 REFERS BOX-2
RELRESTR OBJ-1 B4-1 IN BOX-2 POS
OBJ-1 AMBIG B4-1 BALL-3 BALL-4 ...
PREDINCON OBJ-3 L14-1 SIZE LARGE POS
PREDINCON OBJ-2 L14-1 SIZE LARGE POS
PREDRESTR OBJ-1 L14-1 SIZE LARGE POS
OBJ-1 REFERS BALL-4
REPLY ((THE LARGE GREEN BALL IS NEAR THE SMALL RED BALL AND IN THE UN- RED BOX))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BLOCK-1 BLOCK)
  (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-4 IN BOX-2 POS) (BALL-4 NEAR BALL-3 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

22 INPUT TEXT IS " WHERE IS THE BALL IN THE BOX ON THE RED FLOOR THAT IS RED "
OBJ-1 AMBIG B4-1 BALL-1 BALL-2 ...
OBJ-2 AMBIG B7-1 BOX-1 BOX-2 ...
OBJ-3 AMBIG R10-1 BALL-3 FLOOR-1 ...
OBJ-3 REFERS FLOOR-1
RELRESTR OBJ-2 B7-1 ON FLOOR-1 POS
OBJ-2 REFERS BOX-2
RELRESTR OBJ-1 B4-1 IN BOX-2 POS
OBJ-1 AMBIG B4-1 BALL-3 BALL-4 ...
PREDREDUN OBJ-3 R14-1 COLOR RED POS
PREDRESTR OBJ-1 R14-1 COLOR RED POS
OBJ-1 REFERS BALL-3
REPLY ((THE LARGE GREEN BALL IS NEAR IT))
  ((THE SMALL RED BALL IS IN THE UN- RED BOX))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BLOCK-1 BLOCK)
  (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
```

```
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-4 IN BOX-2 POS) (BALL-4 NEAR BALL-3 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

23 INPUT TEXT IS " THERE IS A BLACK BALL NEAR THE GREEN BALL THAT IS NOT IN THE
  BOX ON THE FLOOR "
ADDING COLOR BLACK (POS) TO BALL-5
ADDING BALL BALL-5
OBJ-2 AMBIG G8-1 BALL-4 BLOCK-1 ...
OBJ-2 REFERS BALL-4
ADDING BALL-5 NEAR BALL-4 (POS)
OBJ-3 AMBIG B15-1 BOX-1 BOX-2 ...
OBJ-4 REFERS FLOOR-1
RELRESTR OBJ-3 B15-1 ON FLOOR-1 POS
OBJ-3 REFERS BOX-2
RELINCON OBJ-2 B9-1 IN BOX-2 NEG
ADDING BALL-5 IN BOX-2 (NEG)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BALL-5 BALL)
  (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BALL-5 COLOR BLACK POS) (BLOCK-1 SIZE LARGE POS)
  (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS)
  (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-4 IN BOX-2 POS) (BALL-4 NEAR BALL-3 POS)
  (BALL-5 NEAR BALL-4 POS) (BALL-5 IN BOX-2 NEG) (BLOCK-1 ON TABLE-1 POS)
  (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS) (BOX-2 ON TABLE-1 NEG)

24 INPUT TEXT IS " THE RED BALL IS NEAR THE GREEN BALL "
OBJ-1 AMBIG R2-1 BALL-3 FLOOR-1 ...
OBJ-1 REFERS BALL-3
OBJ-2 AMBIG G7-1 BALL-4 BLOCK-1 ...
OBJ-2 REFERS BALL-4
RELINCON OBJ-1 B3-1 NEAR BALL-4 POS
ADDING BALL-3 NEAR BALL-4 (POS)
REPLY ((OKAY))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BALL-5 BALL)
  (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BALL-5 COLOR BLACK POS) (BLOCK-1 SIZE LARGE POS)
  (BLOCK-1 COLOR GREEN POS) (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS)
  (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-3 NEAR BALL-4 POS) (BALL-4 IN BOX-2 POS)
  (BALL-4 NEAR BALL-3 POS) (BALL-5 NEAR BALL-4 POS) (BALL-5 IN BOX-2 NEG)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

Z

RUN TIME 4 MIN. 53.8 SEC

EXAM     TRY     FIRE    WMACT  E/F    E/T    T/F
3626     968     634     1813   5.72   3.75   1.53
0.0818   0.304   0.463   0.162  SEC AVG

1107 INSERTS 706 DELETES 296 WARNINGS 15 NEW OBJECTS
MAX :SMPX LENGTH 181
CORE (FREE.FULL): (2211 . 1821) USED (4897 . 735)

(ACTS SAVEOB (CLOSED (MIL46 . OBS)) (CLOSED (MIL46 . TRS)) LOADPS (MILYS . EXP)
   RUN SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY

TRACE
(X20-1
S0-1 G21-1
S1-1 T1-1 G32-1
S1-2 G1-1 N2-1 N9C-1 FS-1 FS-2 FS-3 FS-4 FS-5 FS-6 FS-7 FS-8 FS-9
S1-3 T53-1 N22-1 N33-1 F21-1 F21-2 F21-3 F21-4 F21-5 F21-6 F21-7 F15-1
S1-4 T63-1 P1-1
S1-5 T2-1
S1-6 T4-1 G31-1
S1-7 T7-1 A17-1 F41-1 B23-1 E23-1 F35-1 F13-1
S4-1 B55-1 V17-1 V17-2 V18-1 V18-2 D1-1 D1-2 D1-3 D1-4 D1-5 D2-1 D2-2 D2-3 D11-1
  D11-2 D3-1 D3-2 D3-3 D11-3 D11-4 D2-4 D2-5 D2-6 D2-7 D2-8 D3-4 D3-5 D12-1
  D11-5 D11-6 D2-9 D2-10 D4-1 D4-2 D4-3 D4-4 D4-5 D21-1 D22-1 D23-1 D24-1 D25-1
  D25-2 D26-1 D26-2 D28-1 D28-2 X21-1
S0-2 G21-2
S1-8 T1-2 G32-2
S1-9 G1-2 N2-2 N9C-2 FS-10 FS-11 FS-12 FS-13 FS-14 FS-15 FS-16 FS-17 FS-18
S1-10 T41-1 N22-2 N33-2 F21-8 F21-9 F21-10 F21-11 F21-12 F15-2
S1-11 T31-1 R2-1 R11-1
S1-12 G1-3 N1-1 N98-1 FS-19 FS-20 FS-21 FS-22 FS-23 FS-24 FS-25 FS-26 FS-27
S1-13 T53-2 N22-3 N33-3 F21-13 F21-14 F21-15 F21-16 F21-17 F21-18 F21-19 F15-3
S1-14 T34-1 R2-2 R12-1
S1-15 G1-4 N1-2 N98-2 FS-28 FS-29 FS-30 FS-31 FS-32 FS-33 FS-34 FS-35 FS-36
S1-16 T7-2 A19-1 A1-1 F27-1 F27-2 F27-3 F27-4 F27-5 F27-6 F15-4
S1-17 T50-1 N21-1 N33-4 F21-20 F21-21 F13-2 B1-1 B13-1 E33-1 F31-1 F13-3 B3-1
  B13-2 B13-3 E33-2 F31-2 F31-3 F15-5
S1-18 T63-2 P1-2
S1-19 T1-3 G32-3
S1-20 T21-1 A17-2 F41-2 B28-1 E21-1 B48-1 B28-2 E21-2 B48-2 B23-2 E23-2 F35-2
  F13-4
S4-2 B55-2 V17-3 V17-4 D1-6 D1-7 D1-8 D11-7 D11-8 D2-11 D2-12 D2-13 D3-6 D3-7
  D3-8 D2-14 D12-2 D11-9 D11-10 D4-6 D2-15 D2-16 D4-7 D4-8 D21-2 D22-2 D22-3
  D24-2 X22-1
S0-3 G21-3
S1-21 T1-4 G32-4
S1-22 G1-5 N2-3 N9C-3 FS-37 FS-38 FS-39 FS-40 FS-41 FS-42 FS-43 FS-44 FS-45
S1-23 T41-2 N22-4 N33-5 F21-22 F21-23 F21-24 F21-25 F21-26 F15-6
S1-24 T31-2 R2-3 R11-2
S1-25 G1-6 N1-3 N98-3 FS-46 FS-47 FS-48 FS-49 FS-50 FS-51 FS-52 FS-53 FS-54
S1-26 T53-3 N22-5 N33-6 F21-27 F21-28 F21-29 F21-30 F21-31 F21-32 F21-33 F15-7
S1-27 T34-2 R2-4 R12-2
S1-28 G1-7 N1-4 N98-4 FS-55 FS-56 FS-57 FS-58 FS-59 FS-60 FS-61 FS-62 FS-63
S1-29 T7-3 A19-2 A1-2 F27-7 F27-8 F27-9 F27-10 F27-11 F27-12 F15-8
S1-30 T50-2 N21-2 N33-7 F21-34 F21-35 F13-5 B1-2 B13-4 E33-3 F31-4 F13-6 B3-2
  B13-5 B13-6 E33-4 F31-5 F31-6 F15-9
S1-31 T63-3 P1-3
S1-32 T1-5 G32-5
S1-33 T7-4 A17-3 F41-3 B25-1 E22-1 B41-1 B45-1 B43-1 E23-3 F35-3 F13-7
S4-3 B53-1 B55-3 V17-5 V18-3 D1-9 D1-10 D1-11 D11-11 D11-12 D2-17 D2-18 D2-19
  D3-9 D3-10 D3-11 D2-20 D12-3 D4-9 D11-13 D11-14 D2-21 D2-22 D4-10 D4-11 D25-3
  D26-3 D28-3 D21-3 D22-4 D24-3 B51-1 X23-1
S0-4 G9-1
S1-34 T1-6 G32-6
S1-35 G6-1 N6-1 N9C-4
S1-36 T16-1 A19-3 A5-1
S1-37 T41-3 N21-3 N31-1 N41-1 N51-1 E11-1 E13-1
S1-38 T37-1 R2-5 R11-3
S1-39 G1-8 N1-5 N98-5 FS-64 FS-65 FS-66 FS-67 FS-68 FS-69 FS-70 FS-71 FS-72
S1-40 T18-1 A19-4 A1-3 F27-13 F27-14 F27-15 F27-16 F27-17 F27-18 F27-19 F15-10
S1-41 T41-4 N21-4 N33-8 F21-36 F13-8 B1-3 B11-1 E12-1
S1-42 T63-4 P1-4
S1-43 T2-2
S1-44 T4-2 G31-2
S1-45 T31-3 R1-1 R11-4
S1-46 G1-9 N1-6 N98-6 FS-73 FS-74 FS-75 FS-76 FS-77 FS-78 FS-79 FS-80 FS-81
S1-47 T53-4 N22-6 N33-9 F21-37 F21-38 F21-39 F21-40 F21-41 F21-42 F21-43 F15-1
S1-48 T34-3 R2-6 R12-3
S1-49 G1-10 N1-7 N98-7 FS-82 FS-83 FS-84 FS-85 FS-86 FS-87 FS-88 FS-89 FS-90
S1-50 T50-3 N22-7 N33-10 F21-44 F21-45 F21-46 F21-47 F21-48 F21-49 F21-50 F21-
  F13-9 B1-4 B13-7 E33-5 F31-7 F13-10 B3-3 B18-1 E31-1 B39-1 B11-2 E12-2
S4-4 B55-4 B51-2 B53-2 B53-3 V20-1 X24-1
S0-5 G2-1 N2-4 N90-1 FS-91 FS-92 FS-93 FS-94 FS-95 FS-96 FS-97 FS-98 FS-99
  FS-100
S1-51 T7-5 A19-5 A1-4 F27-20 F27-21 F27-22 F27-23 F27-24 F27-25 F27-26 F15-12
S1-52 T41-5 N21-5 N33-11 F21-52 F21-53 F13-11
S1-53 T1-7 N15-1 B55-5 B59-1 G32-7
S1-54 T37-2 R1-2 R11-5
S1-55 G1-11 N1-8 N98-8 FS-101 FS-102 FS-103 FS-104 FS-105 FS-106 FS-107 FS-108
  FS-109 FS-110
S1-56 T18-2 A19-6 A1-5 F27-27 F27-28 F27-29 F27-30 F27-31 F27-32 F27-33 F27-34
  F15-13
S1-57 T41-6 N21-6 N33-12 F21-54 F13-12 B1-5 B18-2 E31-2 M1-1 E12-3
S4-5 B55-6 B51-3 B53-4 V2-1)

FIRED 102 OUT OF 194 PRODS


              - - - - - - - - - - - - - - - - - - - -


FIFTH SEGMENT TAIL END

25 INPUT TEXT IS " IS THE BALL NEAR THE GREEN BALL IN THE BOX THAT IS NOT ON
  RED TABLE BLACK "
OBJ-1 AMBIG B3-1 BALL-1 BALL-2 ...
OBJ-2 AMBIG G6-1 BALL-4 BLOCK-1 ...
OBJ-2 REFERS BALL-4
```

```
RELRESTR OBJ-1 B3-1 NEAR BALL-4 POS
OBJ-1 AMBIG B3-1 BALL-3 BALL-5 ...
OBJ-3 AMBIG B10-1 BOX-1 BOX-2 ...
OBJ-4 AMBIG R16-1 BALL-3 FLOOR-1 ...
OBJ-4 REFERS TABLE-1
RELRESTR OBJ-3 B10-1 ON TABLE-1 NEG
OBJ-3 PEFERS BOX-2
RELREDUN OBJ-2 B7-1 IN BOX-2 POS
RELRESTR OBJ-1 B7-1 IN BOX-2 POS
OBJ-1 REFERS BALL-3
REPLY ((NO INFOPMATION ON COLOP BLACK))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BALL-5 BALL)
  (BLOCK-1 BLOCK) (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOP BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOP RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOP GREEN POS) (BALL-5 COLOP BLACK POS) (BLOCK-1 SIZE LARGE POS)
  (BLOCK-1 COLOP GREEN POS) (BOX-2 COLOP RED NEG) (FLOOR-1 COLOR RED POS)
  (TABLE-1 COLOP RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-3 NEAR BALL-4 POS) (BALL-4 IN BOX-2 POS)
  (BALL-4 NEAR BALL-3 POS) (BALL-5 NEAR BALL-4 POS) (BALL-5 IN BOX-2 NEG)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

2

RUN TIME 1 MIN. 1.75 SEC

EXAM     TRY     FIRE     WMACT   E/F     E/T     T/F
949      201     173      405     5.49    3.30    1.62
0.0651   0.220   0.357    0.127   SEC AVG

209 INSERTS 196 DELETES 94 WARNINGS 15 NEW OBJECTS
MAX :SMPX LENGTH 80
CORE (FREE.FULL): (5676 . 1560) USED (1432 . 196)

:ACTS SAVEDB (CLOSED (MILS80 . DBS)) (CLOSED (MILS8 . TRS)) RUN SMPXEMPTY

TRACE
((X2S-1
S0-1 T1-1 G10-1 G32-1
S1-1 G1-1 N2-1 N5C-1 FS-1 FS-2 FS-3 FS-4 FS-5 FS-6 FS-7 FS-8 FS-9 FS-10
S1-2 T41-1 N22-1 N33-1 F21-1 F21-2 F21-3 F21-4 F21-5 F15-1
S1-3 T37-1 R2-1 P11-1
S1-4 G1-2 N1-1 N58-1 FS-11 FS-12 FS-13 FS-14 FS-15 FS-16 FS-17 FS-18 FS-19 FS-20
S1-5 T10-1 A19-1 A1-1 F27-1 F27-2 F27-3 F27-4 F27-5 F27-6 F27-7 F27-8 F15-2
S1-6 T41-2 N21-1 N33-2 F21-6 F13-1 B1-1 B13-1 B13-2 E33-1 F31-1 F31-2 F31-3
  F16-3
S1-7 T31-1 R2-2 P12-1
S1-8 G1-3 N1-2 N58-2 FS-21 FS-22 FS-23 FS-24 FS-25 FS-26 FS-27 FS-28 FS-29 FS-30
S1-9 T53-1 N22-2 N33-3 F21-7 F21-8 F21-9 F21-10 F21-11 F21-12 F21-13 F21-14
  F16-4
S1-10 T63-1 P1-1
S1-11 T2-1
S1-12 T4-1 G31-1
S1-13 T34-1 R1-1 R11-2
S1-14 G1-4 N1-3 N58-3 FS-31 FS-32 FS-33 FS-34 FS-35 FS-36 FS-37 FS-38 FS-39
  FS-40
S1-15 T7-1 A19-2 A1-2 F27-9 F27-10 F27-11 F27-12 F27-13 F27-14 F27-15 F15-5
S1-16 T47-1 N21-2 N33-4 F21-15 F21-16 F13-2 B1-2 B13-3 E33-2 F31-4 F13-3 B3-1
  B15-1 E32-1 B31-1 B34-1 E33-3 F31-5 F13-4
S1-17 T16-1 A14-1
S4-1 B51-1 B53-1 B53-2 B55-1 V40-1 V37-1))

FIRED 55 OUT OF 194 PRODS
```

Appendix D. DETAILED TRACE FOR MILIPS TEST2

```
ASSERT (TEST2 'T

TOP LEVEL ASSERT (TEST2 (QUOTE T))
INSERTING (TEST2 T)    X2/

157. X2-1
USING (TEST2 T)
INSERTING (SCANFIN LE-1) (SENTENCE S-1) (ENDMARK LE-1) (ENDMARK RE-1)
 (TEXT 2 (A BLUE BALL IS ON THE TABLE)) (LEFTOF LE-1 A1-1) (EQA A1-1)
 (LEFTOF A1-1 B2-1) (EQBLUE B2-1) (LEFTOF B2-1 B3-1) (EQBALL B3-1)
 (LEFTOF B3-1 I4-1) (EQIS I4-1) (LEFTOF I4-1 O5-1) (EQON O5-1)
 (LEFTOF O5-1 T6-1) (EQTHE T6-1) (LEFTOF T6-1 T7-1) (EQTABLE T7-1)
 (LEFTOF T7-1 RE-1)   S7/S0/

158. S0-2   "SCAN LE"
USING (SCANFIN LE-1) (ENDMARK LE-1) (LEFTOF LE-1 A1-1)
 (TEXT 2 (A BLUE BALL IS ON THE TABLE))

TRACING
2 INPUT TEXT IS " A BLUE BALL IS ON THE TABLE "

INSERTING (SCAN A1-1) (SCANFIN A1-1) (NOT (SCANFIN LE-1)) (TRACING T)   G5T63
 T60T53T50T41/T39T37T31T27T24T16T13/G10G1T10T1/G6T4T2G2 10G02/T21T34/T7T64T57G7
 /

159. G7-2   "A INIT"
USING (SCAN A1-1) (EQA A1-1) (SENTENCE S-1)
INSERTING (INDEFDET A1-1) (GTYPED S-1) (GSD S-1) (WORDEQ A1-1 A)
 (NOT (SCAN A1-1)) (NOT (EQA A1-1))   N2SN5N6/

160. N6-2   "INDEF DET"
USING (INDEFDET A1-1)
INSERTING (NPGCHK A1-1) (DETSEEN A1-1) (CUROBJ OBJ-1 MAIN) (ISINDEF OBJ-1)
 N90/

161. N90-2   "NP GRAM"
USING (NPGCHK A1-1) (LEFTOF LE-1 A1-1) (ENDMARK LE-1)
INSERTING (NOT (NPGCHK A1-1))   N9AA19V12V14B14F53N5/N31A5N23B48B43B38B51B1B33
 V19V17V10R12R11A1N1R24F51F41M15V25G8G1M1M2M5N15V2E8G8E4B4/B1/

162. S1-9   "SCAN ON"
USING (SCANFIN A1-1) (LEFTOF A1-1 B2-1)
INSERTING (SCAN B2-1) (SCANFIN B2-1) (NOT (SCANFIN A1-1))      T47/T57T44G2/G7
 G5T63T60T53T50T41/T39T37T31T27T24T16T13/

163. T13-1   "TAG COLOR3"
USING (SCAN B2-1) (EQBLUE B2-1)
INSERTING (ISAVW B2-1 COLOR BLUE) (WORDEQ B2-1 BLUE) (NOT (SCAN B2-1))
 (NOT (EQBLUE B2-1))   A17A14A19/

164. A19-3   "AV G6"
USING (ISAVW B2-1 COLOR BLUE) (LEFTOF A1-1 B2-1) (DETSEEN A1-1)
INSERTING (ISAV B2-1 COLOR BLUE POS) (NOT (ISAVW B2-1 COLOR BLUE))     A18A6/

165. A5-4   "AV NEW"
USING (ISAV B2-1 COLOR BLUE POS) (CUROBJ OBJ-1 MAIN) (ISINDEF OBJ-1)
INSERTING (NEWAV OBJ-1 COLOR BLUE POS) (OLDAV B2-1)     N81A1N21F41N8AG8E8E4B1/

166. S1-10   "SCAN ON"
USING (SCANFIN B2-1) (LEFTOF B2-1 B3-1)
INSERTING (SCAN B3-1) (SCANFIN B3-1) (NOT (SCANFIN B2-1))      T7T34/T21G8G21
 T2T4G6T1/T10G1/G10G7G2/T44T57T47/T13G5T63T60T53T50T41/

167. T41-1   "TAG NOUN1"
USING (SCAN B3-1) (EQBALL B3-1)
INSERTING (ISNOUNW B3-1 BALL) (WORDEQ B3-1 BALL) (NOT (SCAN B3-1))
 (NOT (EQBALL B3-1))   N22N23/N29/G13N21/

168. N21-3   "N G1"
USING (ISNOUNW B3-1 BALL) (LEFTOF B2-1 B3-1) (ISAV B2-1 COLOR BLUE POS)
INSERTING (ISNOUN B3-1 BALL) (NOT (ISNOUNW B3-1 BALL))  A18N32N31/

169. N31-3   "N INDEF"
USING (ISNOUN B3-1 BALL) (CUROBJ OBJ-1 MAIN) (ISINDEF OBJ-1)
INSERTING (MAKISA B3-1 BALL OBJ-1 MAIN) (ERREF OBJ-1 B3-1)
 (NOT (CUROBJ OBJ-1 MAIN)) (NOT (ISINDEF OBJ-1))      N61/

170. N61-1   "ISA BALL"
USING (MAKISA B3-1 BALL OBJ-1 MAIN)
```

```
INSERTING (ADDAV BALL-1 OBJ-1) (ISA BALL-1 BALL) (CUROBJ BALL-1 MAIN)
 (REFERS BALL-1 BALL-1) (ERRREF BALL-1 B3-1) (NEWOBJ BALL-1)
 (NOT (MAKISA B3-1 BALL OBJ-1 MAIN))  N91/

171.  N91-4  "ADD AVN"
USING (ADDAV BALL-1 OBJ-1) (NEWAV OBJ-1 COLOR BLUE POS)
INSERTING (HASAV BALL-1 COLOR BLUE POS) (NOT (ADDAV BALL-1 OBJ-1))
 (NOT (NEWAV OBJ-1 COLOR BLUE POS))   D11D12B23B41B44B43B29B29V38V39E11/

172.  E11-4  "TRACE AV"
USING (HASAV BALL-1 COLOR BLUE POS)

TRACING
ADDING COLOR BLUE (POS) TO BALL-1


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (TRACING T)  E13/

173.  E13-3  "TRACE ISA"
USING (ISA BALL-1 BALL)

TRACING
ADDING BALL BALL-1


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (TRACING T)  F5F1D4V1DV17V19B33B1N9AB51V12V14B14F53N5/N31A5N33B48B38
 R12R11A1N1B24F51F41M15V25B3B34B53B28B19B18B15F29F23M5M2M1V37V32V31V30B57E8B21
 B11P1R2E8G5E4S1/

174.  S1-11  "SCAN ON"
USING (SCANFIN B3-1) (LEFTOF B3-1 I4-1)
INSERTING (SCAN I4-1) (SCANFIN I4-1) (NOT (SCANFIN B3-1))      T16T24T27T31T37
 T39T50T53T60T63G5T13T47/T57T44G2/G7G1DG1/T10T1/

175.  T1-2   "TAG COP"
USING (SCAN I4-1) (EQIS I4-1) (LEFTOF I4-1 O5-1)
INSERTING (ISCOP I4-1 POS) (WORDEQ I4-1 IS) (NOT (SCAN I4-1)) (NOT (EQIS I4-1))
      A17G18/G17G10N18N8CN15/

176.  N15-2  "NP BDC"
USING (ISCOP I4-1 POS) (SENTENCE S-1) (GSD S-1) (LEFTOF B3-1 I4-1)
INSERTING (NPBOUND I4-1) (NPBOUNDL I4-1)      B57B53B51/B55B58/

177.  B59-2  "NPBND DEL"
USING (NPBOUNDL I4-1)
INSERTING (NOT (NPBOUND I4-1)) (NOT (NPBOUNDL I4-1))   G32/

178.  G32-2  "COP +"
USING (ISCOP I4-1 POS)
WARNING (NEG) NOT UNDER COPSIGN  *.
INSERTING (COPSIGN POS) (NOT (COPSIGN NEG))   R11R1G31/N9AG9E8E4S1/

179.  S1-12  "SCAN ON"
USING (SCANFIN I4-1) (LEFTOF I4-1 O5-1)
INSERTING (SCAN O5-1) (SCANFIN O5-1) (NOT (SCANFIN I4-1))      T41T7T34/

180.  T34-2  "TAG REL2"
USING (SCAN O5-1) (EQON O5-1)
INSERTING (ISRELW O5-1 ON) (WORDEQ O5-1 ON) (NOT (SCAN O5-1)) (NOT (EQON O5-1))
      R1/

181.  R1-2   "REL G1"
USING (ISRELW O5-1 ON) (LEFTOF I4-1 O5-1) (ISCOP I4-1 POS)
INSERTING (ISREL O5-1 ON) (NOT (ISRELW O5-1 ON))      N98R11/

182.  R11-2  "REL NOTE"
USING (ISREL O5-1 ON) (CUROBJ BALL-1 MAIN) (COPSIGN POS)
INSERTING (HASRELN BALL-1 ON POS) (OLDREL O5-1) (NOT (COPSIGN POS))   B3B1/
 R12/E8G9N9AE4S1/

183.  S1-13  "SCAN ON"
USING (SCANFIN O5-1) (LEFTOF O5-1 T6-1)
INSERTING (SCAN T6-1) (SCANFIN T6-1) (NOT (SCANFIN O5-1))      T10G1/

184.  G1-1   "THE"
USING (SCAN T6-1) (EQTHE T6-1) (SENTENCE S-1) (GTYPED S-1)
INSERTING (DEFDET T6-1) (WORDEQ T6-1 THE) (NOT (SCAN T6-1)) (NOT (EQTHE T6-1))
 N22N2/N1/

185.  N1-1   "DEF DET"
USING (DEFDET T6-1) (CUROBJ BALL-1 MAIN)
```

```
INSERTING (NPGCHK T6-1) (DETSEEN T6-1) (DEFFND OBJ-2 T6-1) (CUROBJ OBJ-2 BALL-1)
 (CUROBJP BALL-1 MAIN) (ISDEF OBJ-2) (NOT (CUROBJ BALL-1 MAIN))       N10/N8A
 N8D/N8B/

186.  N8B-2  "NP GRAM"
USING (NPGCHK T6-1) (LEFTOF O5-1 T6-1) (ISREL O5-1 ON)
INSERTING (NOT (NPGCHK T6-1))   A18F8/F8/

187.  F5-1   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BLOCK-1 BLOCK)
INSERTING (FINDPOSS OBJ-2 BLOCK-1) (NOT (DEFFND OBJ-2 T6-1))

188.  F5-2   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA TABLE-1 TABLE)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *.
INSERTING (FINDPOSS OBJ-2 TABLE-1) (NOT (DEFFND OBJ-2 T6-1))   B13B23B27B30B41
 B46B57B44B43B34B33B17V14B31F35F31F27F21F15F13B38B48N33/V10V17V19B1/B51V12B14
 F53N5/N31A5R12/R11A1/N1/B24F51F41M15V25M1M2M5B53M53M51M12M11B3/B58B48B38B35
 M16V48G5E6E4S1/

189.  S1-14  "SCAN ON"
USING (SCANFIN T6-1) (LEFTOF T6-1 T7-1)
INSERTING (SCAN T7-1) (SCANFIN T7-1) (NOT (SCANFIN T6-1))      T34T7T21G8G21T2
 T4G8T1T16T24T27T31T37T39T50T53T60T63T13T47/

190.  T47-2  "TAG NOUN3"
USING (SCAN T7-1) (EQTABLE T7-1)
INSERTING (ISNOUNW T7-1 TABLE) (WORDEQ T7-1 TABLE) (NOT (SCAN T7-1))
 (NOT (EQTABLE T7-1))  G13N29/N23/N22/

191.  N22-1  "N G2"
USING (ISNOUNW T7-1 TABLE) (LEFTOF T6-1 T7-1) (DEFDET T6-1)
INSERTING (ISNOUN T7-1 TABLE) (NOT (ISNOUNW T7-1 TABLE))      R2N33/

192.  N33-1  "N DEF"
USING (ISNOUN T7-1 TABLE) (CUROBJ OBJ-2 BALL-1) (ISDEF OBJ-2)
INSERTING (NRESTR OBJ-2 T7-1 TABLE) (ERRREF OBJ-2 T7-1) F23/F21/

193.  F21-1  "N RESTR"
USING (NRESTR OBJ-2 T7-1 TABLE) (FINDPOSS OBJ-2 BLOCK-1)
INSERTING (OCHK OBJ-2 T7-1) (NOT (NRESTR OBJ-2 T7-1 TABLE))
 (NOT (FINDPOSS OBJ-2 BLOCK-1))      F13/

194.  F13-1  "OBJ FND"
USING (OCHK OBJ-2 T7-1) (FINDPOSS OBJ-2 TABLE-1)

TRACING
OBJ-2 REFERS TABLE-1


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (REFERS OBJ-2 TABLE-1) (TRACING T) (NOT (OCHK OBJ-2 T7-1))
 (NOT (FINDPOSS OBJ-2 TABLE-1))      V10V17V19V30V31V39V38M1M2M5B15B18B18B28
 B29B33B35B53B34B3/B1/

195.  B1-2   "DEF REF"
USING (REFERS OBJ-2 TABLE-1) (CUROBJ OBJ-2 BALL-1) (HASRELN BALL-1 ON POS)
 (ERRREF BALL-1 B3-1)
INSERTING (RELRESTRCHK BALL-1 B3-1 ON TABLE-1 POS) (CUROBJP OBJ-2 BALL-1)
 (OLDREF OBJ-2)      B13B19/B18/B17B15/B14B11/

196.  B11-2  "REL RCHK NEW"
USING (RELRESTRCHK BALL-1 B3-1 ON TABLE-1 POS) (NEWOBJ BALL-1)
INSERTING (HASREL BALL-1 ON TABLE-1 POS)
 (NOT (RELRESTRCHK BALL-1 B3-1 ON TABLE-1 POS))      V30V31B13B31B34B33B19
 B19V17E12/

197.  E12-2  "TRACE REL"
USING (HASREL BALL-1 ON TABLE-1 POS)

TRACING
ADDING BALL-1 ON TABLE-1 (POS)


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (TRACING T)  V10B36B46B38B4B83/M11M12M51M53B43B44B53M5M2M1B55B43B39
 B35M16V48B78F29F23V37V32E8B57A1N3IP1E8G5N9AE4S1/S4/

198.  S4-2   "SCANFIN"
USING (SCANFIN T7-1) (LEFTOF T7-1 RE-1) (ENDMARK RE-1) (SENTENCE S-1)
INSERTING (NPBOUND RE-1) (SENTBOUND S-1) (NOT (SCANFIN T7-1))  B81/

199.  B81-2  "NPBND UNDO"
```

```
USING (NPBOUND RE-1) (CUROBJ OBJ-2 BALL-1) (REFERS OBJ-2 TABLE-1)
INSERTING (NOT (CUROBJ OBJ-2 BALL-1))  B46B34B53/

1100. B53-2  "NPEND UNDOP"
USING (NPBOUND RE-1) (CUROBJP OBJ-2 BALL-1) (REFERS OBJ-2 TABLE-1)
INSERTING (NOT (CUROBJP OBJ-2 BALL-1)) B57B55/

1101. B55-2  "NPEND REDO"
USING (NPBOUND RE-1) (CUROBJP BALL-1 MAIN)
INSERTING (CUROBJ BALL-1 MAIN)  V14V12V19V17V10B3B B43B48N33/N8AB33B1/B51/B14F53
 N5/N3 1A5R12/R11A1/A11/B24F51F41M19V2 8V36V39V31V30V2/

1102. V2-2   "REPLY SD"
USING (SENTBOUND S-1) (GSD S-1)
INSERTING (REPLY (OKAY))       V5V49V48V46V44V42V40V37V32V20

REPLY ((OKAY))

ISA (BALL-1 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
 (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BLOCK-1 ON TABLE-1 POS)

CUROBJ (BALL-1 MAIN)
CUROBJP (BALL-1 MAIN)
DEFDET (T6-1)
DETSEEN (A1-1) (T6-1)
ENDMARK (LE-1) (RE-1)
ERRREF (BALL-1 B3-1) (OBJ-1 B3-1) (OBJ-2 T7-1)
GSD (S-1)
GTYPED (S-1)
HASAV (BALL-1 COLOR BLUE POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
 (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BLOCK-1 ON TABLE-1 POS)
HASRELN (BALL-1 ON POS)
INDEFDET (A1-1)
ISA (BALL-1 BALL) (BLOCK-1 BLOCK) (TABLE-1 TABLE)
ISAV (B2-1 COLOR BLUE POS)
ISCOP (I4-1 POS)
ISDEF (OBJ-2)
ISNOUN (B3-1 BALL) (T7-1 TABLE)
ISREL (O5-1 ON)
LEFTOF (A1-1 B2-1) (B2-1 B3-1) (B3-1 I4-1) (I4-1 O5-1) (LE-1 A1-1) (O5-1 T6-1)
 (T6-1 T7-1) (T7-1 RE-1)
NEWOBJ (BALL-1)
NPBOUND (RE-1)
OLDAV (B2-1)
OLDREF (OBJ-2)
OLDREL (O5-1)
REFERS (BALL-1 BALL-1) (OBJ-2 TABLE-1)
REPLY ((OKAY))
SENTBOUND (S-1)
SENTENCE (S-1)
TEST2 (T)
TEXT (2 (A BLUE BALL IS ON THE TABLE))
TRACING (T)
WORDEQ (A1-1 A) (B2-1 BLUE) (B3-1 BALL) (I4-1 IS) (O5-1 ON) (T6-1 THE)
 (T7-1 TABLE)
ASSERT (TEST22 'T

TOP LEVEL ASSERT (TEST22 (QUOTE T))
INSERTING (TEST22 T)   X22/

1249. X22-1
USING (TEST22 T)
INSERTING (SCANFIN LE-1) (SENTENCE S-1) (ENDMARK LE-1) (ENDMARK RE-1)
 (TEXT 22 (WHERE IS THE BALL IN THE BOX ON THE RED FLOOR THAT IS RED))
 (LEFTOF LE-1 W1-1) (EQWHERE W1-1) (LEFTOF W1-1 I2-1) (EQIS I2-1)
 (LEFTOF I2-1 T3-1) (EQTHE T3-1) (LEFTOF T3-1 B4-1) (EQBALL B4-1)
 (LEFTOF B4-1 I5-1) (EQIN I5-1) (LEFTOF I5-1 T6-1) (EQTHE T6-1)
 (LEFTOF T6-1 B7-1) (EQBOX B7-1) (LEFTOF B7-1 O8-1) (EQON O8-1)
 (LEFTOF O8-1 T9-1) (EQTHE T9-1) (LEFTOF T9-1 R10-1) (EQRED R10-1)
 (LEFTOF R10-1 F11-1) (EQFLOOR F11-1) (LEFTOF F11-1 T12-1) (EQTHAT T12-1)
 (LEFTOF T12-1 I13-1) (EQIS I13-1) (LEFTOF I13-1 R14-1) (EQRED R14-1)
 (LEFTOF R14-1 RE-1)   S7/S4/S0/

1250. S0-3   "SCAN LE"
USING (SCANFIN LE-1) (ENDMARK LE-1) (LEFTOF LE-1 W1-1)
 (TEXT 22 (WHERE IS THE BALL IN THE BOX ON THE RED FLOOR THAT IS RED))

TRACING
22 INPUT TEXT IS " WHERE IS THE BALL IN THE BOX ON THE RED FLOOR THAT IS RED "
```

```
INSERTING (SCAN W1-1) (SCANFIN W1-1) (NOT (SCANFIN LE-1)) (TRACING T)   T1/T80/
 G1T83/T31/G8T2T5702/G700T34/T83/T7/T4621/

1251. G21-3  "WHERE"
USING (SCAN W1-1) (EQWHERE W1-1) (SENTENCE S-1)
INSERTING (GSQWR S-1) (GTYPED S-1) (WORDEQ W1-1 WHERE) (NOT (SCAN W1-1))
 (NOT (EQWHERE W1-1))  V19V17G8G1G5E[illegible]/B1/

1252. S1-21  "SCAN ON"
USING (SCANFIN W1-1) (LEFTOF W1-1 I2-1)
INSERTING (SCAN I2-1) (SCANFIN I2-1) (NOT (SCANFIN W1-1))      G10T5T7T10T80T13
 T30T24T16T47T44T27T41/G5T21T4T7/T63/T34/G8G7G2/T57T2T31/T[illegible]/B1/T80/T1/

1253. T1-4   "TAG COP"
USING (SCAN I2-1) (EQIS I2-1) (LEFTOF I2-1 T3-1)
INSERTING (ISCOP I2-1 POS) (WORDEQ I2-1 IS) (NOT (SCAN I2-1)) (NOT (EQIS I2-1))
     N1GN15G10G17G18/B1NBCA17G32/

1254. G32-4  "COP +"
USING (ISCOP I2-1 POS)
WARNING (NEG) NOT UNDER COPSIGN  *-
INSERTING (COPSIGN POS) (NOT (COPSIGN NEG))    R11G31/N[illegible]481/

1255. S1-22  "SCAN ON"
USING (SCANFIN I2-1) (LEFTOF I2-1 T3-1)
INSERTING (SCAN T3-1) (SCANFIN T3-1) (NOT (SCANFIN I2-1))      G8G2102/T61/T27
 T44T47T16T24T39T13T80T10T37T1/G10G5T21T4T7/T63/T34/G8G7T57T2T31/T83/B1/

1256. G1-5   "THE"
USING (SCAN T3-1) (EQTHE T3-1) (SENTENCE S-1) (GTYPED S-1)
INSERTING (DEFDET T3-1) (WORDEQ T3-1 THE) (NOT (SCAN T3-1)) (NOT (EQTHE T3-1))
 N22N1N2/

1257. N2-3   "DEF DET"
USING (DEFDET T3-1)
INSERTING (NPGCHK T3-1) (DETSEEN T3-1) (DEFFND OBJ-1 T3-1) (CUROBJP OBJ-1 MAIN)
 (CUROBJ OBJ-1 MAIN) (ISDEF OBJ-1)    N10/N80/N[illegible]C/

1258. N9C-3  "NP GRAM"
USING (NPGCHK T3-1) (LEFTOF I2-1 T3-1) (ISCOP I2-1 POS)
INSERTING (NOT (NPGCHK T3-1))   A19F5/

1259. F5-37  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA BALL-1 BALL)
INSERTING (FINDPOSS OBJ-1 BALL-1) (NOT (DEFFND OBJ-1 T3-1))

1260. F5-38  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA BALL-2 BALL)
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 BALL-2) (NOT (DEFFND OBJ-1 T3-1))

1261. F5-39  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA BALL-3 BALL)
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 BALL-3) (NOT (DEFFND OBJ-1 T3-1))

1262. F5-40  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA BALL-4 BALL)
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 BALL-4) (NOT (DEFFND OBJ-1 T3-1))

1263. F5-41  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA BLOCK-1 BLOCK)
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 BLOCK-1) (NOT (DEFFND OBJ-1 T3-1))

1264. F5-42  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA BOX-1 BOX)
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 BOX-1) (NOT (DEFFND OBJ-1 T3-1))

1265. F5-43  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA BOX-2 BOX)
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 BOX-2) (NOT (DEFFND OBJ-1 T3-1))

1266. F5-44  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA FLOOR-1 FLOOR)
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 FLOOR-1) (NOT (DEFFND OBJ-1 T3-1))

1267. F5-45  "DEF FIND"
USING (DEFFND OBJ-1 T3-1) (ISA TABLE-1 TABLE)
```

```
WARNING (OBJ-1 T3-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-1 TABLE-1) (NOT (DEFFND OBJ-1 T3-1))   B23B57B27B13F31
 F21F13B34B33V14B38B4B43B41B17F27F1[illegible]
 B59B45B3SM1BVA[illegible]33A1F53V10V17V1[illegible]/F51V2SGS
 E6E4S1/

1268. S1-23   "SCAN ON"
USING (SCANFIN T3-1) (LEFTOF T3-1 B4-1)
INSERTING (SCAN B4-1) (SCANFIN B4-1) (NOT (SCANFIN T3-1))     T50/T53/T31/T2
 T57G7G9T34/T63/T7/T4T21G5G10T1/T37T10T60T13T39T24T10T47T44T27T41/

1269. T41-2   "TAG NOUN1"
USING (SCAN B4-1) (EQBALL B4-1)
INSERTING (ISNOUNW B4-1 BALL) (WORDEQ B4-1 BALL) (NOT (SCAN B4-1))
 (NOT (EQBALL B4-1))  N22/

1270. N22-4   "N G2"
[illegible]NOUNW B4-1 BALL) (LEFTOF T3-1 B4-1) (DEFDET T3-1)
INSERTING (ISNOUN B4-1 BALL) (NOT (ISNOUNW B4-1 BALL))  A14N33/

1271. N33-9   "N DEF"
USING (ISNOUN B4-1 BALL) (CUROBJ OBJ-1 MAIN) (ISDEF OBJ-1)
INSERTING (NRESTR OBJ-1 B4-1 BALL) (ERRREF OBJ-1 B4-1)  F21/

1272. F21-22  "N RESTR"
USING (NRESTR OBJ-1 B4-1 BALL) (FINDPOSS OBJ-1 BLOCK-1)
INSERTING (OCHK OBJ-1 B4-1) (NOT (NRESTR OBJ-1 B4-1 BALL))
 (NOT (FINDPOSS OBJ-1 BLOCK-1))

1273. F21-23  "N RESTR"
USING (NRESTR OBJ-1 B4-1 BALL) (FINDPOSS OBJ-1 BOX-1)
WARNING (OBJ-1 B4-1) ALREADY UNDER OCHK  *.
WARNING (OBJ-1 B4-1 BALL) NOT UNDER NRESTR  *-
INSERTING (OCHK OBJ-1 B4-1) (NOT (NRESTR OBJ-1 B4-1 BALL))
 (NOT (FINDPOSS OBJ-1 BOX-1))

1274. F21-24  "N RESTR"
USING (NRESTR OBJ-1 B4-1 BALL) (FINDPOSS OBJ-1 BOX-2)
WARNING (OBJ-1 B4-1) ALREADY UNDER OCHK  *.
WARNING (OBJ-1 B4-1 BALL) NOT UNDER NRESTR  *-
INSERTING (OCHK OBJ-1 B4-1) (NOT (NRESTR OBJ-1 B4-1 BALL))
 (NOT (FINDPOSS OBJ-1 BOX-2))

1275. F21-25  "N RESTR"
USING (NRESTR OBJ-1 B4-1 BALL) (FINDPOSS OBJ-1 FLOOR-1)
WARNING (OBJ-1 B4-1) ALREADY UNDER OCHK  *.
WARNING (OBJ-1 B4-1 BALL) NOT UNDER NRESTR  *-
INSERTING (OCHK OBJ-1 B4-1) (NOT (NRESTR OBJ-1 B4-1 BALL))
 (NOT (FINDPOSS OBJ-1 FLOOR-1))

1276. F21-26  "N RESTR"
USING (NRESTR OBJ-1 B4-1 BALL) (FINDPOSS OBJ-1 TABLE-1)
WARNING (OBJ-1 B4-1) ALREADY UNDER OCHK  *.
WARNING (OBJ-1 B4-1 BALL) NOT UNDER NRESTR  *-
INSERTING (OCHK OBJ-1 B4-1) (NOT (NRESTR OBJ-1 B4-1 BALL))
 (NOT (FINDPOSS OBJ-1 TABLE-1))      F19/

1277. F15-6   "OBJ MULT"
USING (OCHK OBJ-1 B4-1) (FINDPOSS OBJ-1 BALL-1) (FINDPOSS OBJ-1 BALL-2)

TRACING
OBJ-1 AMBIG B4-1 BALL-1 BALL-2 _


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (TRACING T) (NOT (OCHK OBJ-1 B4-1))  B57E8B38I[illegible]G13N8AE8G5E4S1/

1278. S1-24   "SCAN ON"
USING (SCANFIN B4-1) (LEFTOF B4-1 I5-1)
INSERTING (SCAN I5-1) (SCANFIN I5-1) (NOT (SCANFIN B4-1))     G1/G8G21G2/T41
 T50/T53/T31/

1279. T31-2   "TAG REL1"
USING (SCAN I5-1) (EQIN I5-1)
INSERTING (ISRELW I5-1 IN) (WORDEQ I5-1 IN) (NOT (SCAN I5-1)) (NOT (EQIN I5-1))
     R5/R3R1/R2/

1280. R2-3    "REL G2"
USING (ISRELW I5-1 IN) (LEFTOF B4-1 I5-1) (ISNOUN B4-1 BALL)
INSERTING (ISREL I5-1 IN) (NOT (ISRELW I5-1 IN))      R11/

1281. R11-2   "REL NOTE"
USING (ISREL I5-1 IN) (CUROBJ OBJ-1 MAIN) (COPSIGN POS)
INSERTING (HASRELN OBJ-1 IN POS) (OLDREL I5-1) (NOT (COPSIGN POS))     [illegible]
 R12/G5E8N8AE4S1/

1282. S1-25   "SCAN ON"
USING (SCANFIN I5-1) (LEFTOF I5-1 T6-1)
INSERTING (SCAN T6-1) (SCANFIN T6-1) (NOT (SCANFIN I5-1))      T27T46T67T10T20
 T30T13T60T10T37T1/G10G9T21T4T7/T63/T34/[illegible]/T50/T41G2/[illegible]1/

1283. G1-6    "THE"
USING (SCAN T6-1) (EQTHE T6-1) (SENTENCE S-1) (STYPED S-1)
INSERTING (DEFDET T6-1) (WORDEQ T6-1 THE) (NOT (SCAN T6-1)) (NOT (EQTHE T6-1))
 N1/

1284. N1-3    "DEF DET"
USING (DEFDET T6-1) (CUROBJ OBJ-1 MAIN)
WARNING (OBJ-1 MAIN) ALREADY UNDER CUROBJP  *.
INSERTING (NPGCHK T6-1) (DETSEEN T6-1) (DEFFND OBJ-2 T6-1) (CUROBJ OBJ-2 OBJ-1)
 (CUROBJP OBJ-1 MAIN) (ISDEF OBJ-2) (NOT (CUROBJ OBJ-1 MAIN))  [illegible]/N8C/A10/
 N98/

1285. N98-3   "NP GRAM"
USING (NPGCHK T6-1) (LEFTOF I5-1 T6-1) (ISREL I5-1 IN)
INSERTING (NOT (NPGCHK T6-1))  A18F9/

1286. F5-46   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BALL-1 BALL)
INSERTING (FINDPOSS OBJ-2 BALL-1) (NOT (DEFFND OBJ-2 T6-1))

1287. F5-47   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BALL-2 BALL)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 BALL-2) (NOT (DEFFND OBJ-2 T6-1))

1288. F5-48   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BALL-3 BALL)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 BALL-3) (NOT (DEFFND OBJ-2 T6-1))

1289. F5-49   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BALL-4 BALL)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 BALL-4) (NOT (DEFFND OBJ-2 T6-1))

1290. F5-50   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BLOCK-1 BLOCK)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 BLOCK-1) (NOT (DEFFND OBJ-2 T6-1))

1291. F5-51   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BOX-1 BOX)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 BOX-1) (NOT (DEFFND OBJ-2 T6-1))

1292. F5-52   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA BOX-2 BOX)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 BOX-2) (NOT (DEFFND OBJ-2 T6-1))

1293. F5-53   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA FLOOR-1 FLOOR)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 FLOOR-1) (NOT (DEFFND OBJ-2 T6-1))

1294. F5-54   "DEF FIND"
USING (DEFFND OBJ-2 T6-1) (ISA TABLE-1 TABLE)
WARNING (OBJ-2 T6-1) NOT UNDER DEFFND  *-
INSERTING (FINDPOSS OBJ-2 TABLE-1) (NOT (DEFFND OBJ-2 T6-1))    F39B44B3IF19F27
 B17B41B43B46B36V14B33B34F13B27B27B13F31B57F21N33/A1F53V10V17V1[illegible]
 N31A5V12B14M15F41B24B12/R11N1/F51V2SM12M53M11M5I[illegible]
 N22N2/E6G5E4S1/

1295. S1-26   "SCAN ON"
USING (SCANFIN T6-1) (LEFTOF T6-1 B7-1)
INSERTING (SCAN B7-1) (SCANFIN B7-1) (NOT (SCANFIN T6-1))      T31G8G21G2/T41
 T50/T53/

1296. T53-3   "TAG NOUNS"
USING (SCAN B7-1) (EQBOX B7-1)
INSERTING (ISNOUNW B7-1 BOX) (WORDEQ B7-1 BOX) (NOT (SCAN B7-1))
 (NOT (EQBOX B7-1))   N21N23N29/G13N22/

1297. N22-9   "N G2"
USING (ISNOUNW B7-1 BOX) (LEFTOF T6-1 B7-1) (DEFDET T6-1)
```

```
INSERTING (ISNOUN B7-1 BOX) (NOT (ISNOUNW B7-1 BOX))   N3[illegible]/

1298. N33-6  "N DEF"
USING (ISNOUN B7-1 BOX) (CUROBJ OBJ-2 OBJ-1) (ISDEF OBJ-2)
INSERTING (NRESTR OBJ-2 B7-1 BOX) (ENNREF OBJ-2 B7-1)  F21/

1299. F21-27  "N RESTR"
USING (NRESTR OBJ-2 B7-1 BOX) (FINDPOSS OBJ-2 BALL-1)
INSERTING (OCHK OBJ-7 B7-1) (NOT (NRESTR OBJ-2 B7-1 BOX))
  (NOT (FINDPOSS OBJ-2 BALL-1)))

1300. F21-28  "N RESTR"
USING (NRESTR OBJ-2 B7-1 BOX) (FINDPOSS OBJ-2 BALL-2)
WARNING (OBJ-2 B7-1) ALREADY UNDER OCHK  ■•
WARNING (OBJ-2 B7-1 BOX) NOT UNDER NRESTR  ■-
INSERTING (OCHK OBJ-2 B7-1) (NOT (NRESTR OBJ-2 B7-1 BOX))
  (NOT (FINDPOSS OBJ-2 BALL-2))

1301. F21-29  "N RESTR"
USING (NRESTR OBJ-2 B7-1 BOX) (FINDPOSS OBJ-2 BALL-3)
WARNING (OBJ-2 B7-1) ALREADY UNDER OCHK  ■•
WARNING (OBJ-2 B7-1 BOX) NOT UNDER NRESTR  ■-
INSERTING (OCHK OBJ-7 B7-1) (NOT (NRESTR OBJ-2 B7-1 BOX))
  (NOT (FINDPOSS OBJ-2 BALL-3)))

1302. F21-30  "N RESTR"
USING (NRESTR OBJ-7 B7-1 BOX) (FINDPOSS OBJ-2 BALL-4)
WARNING (OBJ-2 B7-1) ALREADY UNDER OCHK  ■•
WARNING (OBJ-2 B7-1 BOX) NOT UNDER NRESTR  ■-
INSERTING .OCHK OBJ-2 B7-1) (NOT (NRESTR OBJ-2 B7-1 BOX))
  (NOT (FINDPOSS OBJ-2 BALL-4))

1303. F21-31  "N RESTR"
USING (NRESTR OBJ-7 B7-1 BOX) (FINDPOSS OBJ-2 BLOCK-1)
WARNING (OBJ-2 B7-1) ALREADY UNDER OCHK  ■•
WARNING (OBJ-2 B7-1 BOX) NOT UNDER NRESTR  ■-
INSERTING (OCHK OBJ-2 B7-1) (NOT (NRESTR OBJ-2 B7-1 BOX))
  (NOT (FINDPOSS OBJ-2 BLOCK-1))

1304. F21-32  "N RESTR"
USING (NRESTR OBJ-2 B7-1 BOX) (FINDPOSS OBJ-2 FLOOR-1)
WARNING (OBJ-2 B7-1) ALREADY UNDER OCHK  ■•
WARNING (OBJ-2 B7-1 BOX) NOT UNDER NRESTR  ■-
INSERTING (OCHK OBJ-2 B7-1) (NOT (NRESTR OBJ-2 B7-1 BOX))
  (NOT (FINDPOSS OBJ-2 FLOOR-1))

1305. F21-33  "N RESTR"
USING (NRESTR OBJ-2 B7-1 BOX) (FINDPOSS OBJ-2 TABLE-1)
WARNING (OBJ-2 B7-1) ALREADY UNDER OCHK  ■•
WARNING (OBJ-2 B7-1 BOX) NOT UNDER NRESTR  ■-
INSERTING (OCHK OBJ-2 B7-1) (NOT (NRESTR OBJ-2 B7-1 BOX))
  (NOT (FINDPOSS OBJ-2 TABLE-1))      F13/F11/F19/

1306. F15-7  "OBJ MULT"
USING (OCHK OBJ-2 B7-1) (FINDPOSS OBJ-2 BOX-1) (FINDPOSS OBJ-2 BOX-2)

TRACING
OBJ-2 AMBIG B7-1 BOX-1 BOX-2 _


WARNING (T) ALREADY UNDER TRACING  ■•
INSERTING (TRACING T) (NOT (OCHK OBJ-2 B7-1))   B57[illegible]1P1G5[illegible]4S1/

1307. S1-27  "SCAN ON"
USING (SCANFIN B7-1) (LEFTOF B7-1 OB-1)
INSERTING (SCAN OB-1) (SCANFIN OB-1) (NOT (SCANFIN B7-1))      G1/T27T44T47T16
  T24T39T13T60T10T37T1/G10G5T21T4T7/T63/T34/

1308. T34-2  "TAG REL2"
USING (SCAN OB-1) (EQON OB-1)
INSERTING (ISRELW OB-1 ON) (WORDEQ OB-1 ON) (NOT (SCAN OB-1)) (NOT (EQON OB-1))
     R1/R3R5/R2/

1309. R2-4  "REL G2"
USING (ISRELW OB-1 ON) (LEFTOF B7-1 OB-1) (ISNOUN B7-1 BOX)
INSERTING (ISREL OB-1 ON) (NOT (ISRELW OB-1 ON))      R12/

1310. R12-2  "REL NOTE2"
USING (ISREL OB-1 ON) (CUROBJ OBJ-2 OBJ-1)
INSERTING (HASRELN OBJ-2 ON POS) (OLDREL OB-1)  B3B1N[illegible]1N[illegible]4S1/

1311. S1-28  "SCAN ON"
USING (SCANFIN OB-1) (LEFTOF OB-1 T9-1)
INSERTING (SCAN T9-1) (SCANFIN T9-1) (NOT (SCANFIN OB-1))      T[illegible]/
  T61T50/T2T57[illegible]T34G1/

1312. D1-7  "THE"
USING (SCAN T9-1) (EQTHE T9-1) (SENTENCE S-1) (STYPED S-1)
INSERTING (DEFDET T9-1) (WORDEQ T9-1 THE) (NOT (SCAN T9-1)) (NOT (EQTHE T9-1))
  N[illegible]1/

1313. N1-4  "DEF DET"
USING (DEFDET T9-1) (CUROBJ OBJ-2 OBJ-1)
INSERTING (NPGCHK T9-1) (DETSEEN T9-1) (DEFFND OBJ-3 T9-1) (CUROBJ OBJ-3 OBJ-2)
  (CUROBJP OBJ-2 OBJ-1) (ISDEF OBJ-3) (NOT (CUROBJ OBJ-2 OBJ-1))      [illegible]/

1314. N58-4  "NP GRAM"
USING (NPGCHK T9-1) (LEFTOF OB-1 T9-1) (ISREL OB-1 ON)
INSERTING (NOT (NPGCHK T9-1))  A1[illegible]5/

1315. F5-55  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA BALL-1 BALL)
INSERTING (FINDPOSS OBJ-3 BALL-1) (NOT (DEFFND OBJ-3 T9-1))

1316. F5-56  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA BALL-2 BALL)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 BALL-2) (NOT (DEFFND OBJ-3 T9-1))

1317. F5-57  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA BALL-3 BALL)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 BALL-3) (NOT (DEFFND OBJ-3 T9-1))

1318. F5-58  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA BALL-4 BALL)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 BALL-4) (NOT (DEFFND OBJ-3 T9-1))

1319. F5-59  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA BLOCK-1 BLOCK)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 BLOCK-1) (NOT (DEFFND OBJ-3 T9-1))

1320. F5-60  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA BOX-1 BOX)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 BOX-1) (NOT (DEFFND OBJ-3 T9-1))

1321. F5-61  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA BOX-2 BOX)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 BOX-2) (NOT (DEFFND OBJ-3 T9-1))

1322. F5-62  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA FLOOR-1 FLOOR)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 FLOOR-1) (NOT (DEFFND OBJ-3 T9-1))

1323. F5-63  "DEF FIND"
USING (DEFFND OBJ-3 T9-1) (ISA TABLE-1 TABLE)
WARNING (OBJ-3 T9-1) NOT UNDER DEFFND  ■-
INSERTING (FINDPOSS OBJ-3 TABLE-1) (NOT (DEFFND OBJ-3 T9-1))   F21B57F31B13B27
  B23F13B34B33V14B36B4B84384 1B17F27F15B31B44F39N33/A1F53V10V17V1[illegible]1B9 1N5
  N3 1A5V17B14M19F41B24R12/R11N1/F51V25M12M53M11M51[illegible]16V4[illegible]
  N2/G5[illegible]4S1/

1324. S1-29  "SCAN ON"
USING (SCANFIN T9-1) (LEFTOF T9-1 R10-1)
INSERTING (SCAN R10-1) (SCANFIN R10-1) (NOT (SCANFIN T9-1))     T63/T7/

1325. T7-3  "TAG COLOR1"
USING (SCAN R10-1) (EQRED R10-1)
INSERTING (ISAVW R10-1 COLOR RED) (WORDEQ R10-1 RED) (NOT (SCAN R10-1))
  (NOT (EQRED R10-1))  A19/

1326. A19-2  "AV GB"
USING (ISAVW R10-1 COLOR RED) (LEFTOF T9-1 R10-1) (DETSEEN T9-1)
INSERTING (ISAV R10-1 COLOR RED POS) (NOT (ISAVW R10-1 COLOR RED))     A1/

1327. A1-2  "AV NFND"
USING (ISAV R10-1 COLOR RED POS) (CUROBJ OBJ-3 OBJ-2) (ISDEF OBJ-3)
INSERTING (AVRESTR OBJ-3 R10-1 COLOR RED POS) (OLDAV R10-1)     F27/

1328. F27-7  "AV RESTR"
USING (AVRESTR OBJ-3 R10-1 COLOR RED POS) (FINDPOSS OBJ-3 BALL-1)
```

```
INSERTING (OCHK OBJ-3 R10-1) (NOT (FINDPOSS OBJ-3 BALL-1))

1329. F27-8  "AV RESTR"
USING (AVRESTR OBJ-3 R10-1 COLOR RED POS) (FINDPOSS OBJ-3 BALL-2)
WARNING (OBJ-3 R10-1) ALREADY UNDER OCHK  *.
INSERTING (OCHK OBJ-3 R10-1) (NOT (FINDPOSS OBJ-3 BALL-2))

1330. F27-9  "AV RESTR"
USING (AVRESTR OBJ-3 R10-1 COLOR RED POS) (FINDPOSS OBJ-3 BALL-4)
WARNING (OBJ-3 R10-1) ALREADY UNDER OCHK  *.
INSERTING (OCHK OBJ-3 R10-1) (NOT (FINDPOSS OBJ-3 BALL-4))

1331. F27-10 "AV RESTR"
USING (AVRESTR OBJ-3 R10-1 COLOR RED POS) (FINDPOSS OBJ-3 BLOCK-1)
WARNING (OBJ-3 R10-1) ALREADY UNDER OCHK  *.
INSERTING (OCHK OBJ-3 R10-1) (NOT (FINDPOSS OBJ-3 BLOCK-1))

1332. F27-11 "AV RESTR"
USING (AVRESTR OBJ-3 R10-1 COLOR RED POS) (FINDPOSS OBJ-3 BOX-1)
WARNING (OBJ-3 R10-1) ALREADY UNDER OCHK  *.
INSERTING (OCHK OBJ-3 R10-1) (NOT (FINDPOSS OBJ-3 BOX-1))

1333. F27-12 "AV RESTR"
USING (AVRESTR OBJ-3 R10-1 COLOR RED POS) (FINDPOSS OBJ-3 BOX-2)
WARNING (OBJ-3 R10-1) ALREADY UNDER OCHK  *.
INSERTING (OCHK OBJ-3 R10-1) (NOT (FINDPOSS OBJ-3 BOX-2))      F15/

1334. F15-8  "OBJ MULT"
USING (OCHK OBJ-3 R10-1) (FINDPOSS OBJ-3 BALL-3) (FINDPOSS OBJ-3 FLOOR-1)

TRACING
OBJ-3 AMBIG R10-1 BALL-3 FLOOR-1 _


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (TRACING T) (NOT (OCHK OBJ-3 R10-1))  F29A5A1BN21F41N8AE8G8E4S1/

1335. S1-30  "SCAN ON"
USING (SCANFIN R10-1) (LEFTOF R10-1 F11-1)
INSERTING (SCAN F11-1) (SCANFIN F11-1) (NOT (SCANFIN R10-1))   G1T53T31G0G21G2
 T41T50/

1336. T50-2  "TAG NOUNS"
USING (SCAN F11-1) (EQFLOOR F11-1)
INSERTING (ISNOUNW F11-1 FLOOR) (WORDEQ F11-1 FLOOR) (NOT (SCAN F11-1))
 (NOT (EQFLOOR F11-1)) G13N29/N23N21/

1337. N21-2  "N G1"
USING (ISNOUNW F11-1 FLOOR) (LEFTOF R10-1 F11-1) (ISAV R10-1 COLOR RED POS)
INSERTING (ISNOUN F11-1 FLOOR) (NOT (ISNOUNW F11-1 FLOOR))    A16R2N31N33/

1338. N33-7  "N DEF"
USING (ISNOUN F11-1 FLOOR) (CUROBJ OBJ-3 OBJ-2) (ISDEF OBJ-3)
INSERTING (NRESTR OBJ-3 F11-1 FLOOR) (ERRREF OBJ-3 F11-1)      F21/

1339. F21-34 "N RESTR"
USING (NRESTR OBJ-3 F11-1 FLOOR) (FINDPOSS OBJ-3 BALL-3)
INSERTING (OCHK OBJ-3 F11-1) (NOT (NRESTR OBJ-3 F11-1 FLOOR))
 (NOT (FINDPOSS OBJ-3 BALL-3))

1340. F21-39 "N RESTR"
USING (NRESTR OBJ-3 F11-1 FLOOR) (FINDPOSS OBJ-3 TABLE-1)
WARNING (OBJ-3 F11-1) ALREADY UNDER OCHK  *.
WARNING (OBJ-3 F11-1 FLOOR) NOT UNDER NRESTR  *.
INSERTING (OCHK OBJ-3 F11-1) (NOT (NRESTR OBJ-3 F11-1 FLOOR))
 (NOT (FINDPOSS OBJ-3 TABLE-1))      F15/F13/

1341. F13-5  "OBJ FND"
USING (OCHK OBJ-3 F11-1) (FINDPOSS OBJ-3 FLOOR-1)

TRACING
^BJ-3 REFERS FLOOR-1


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (REFERS OBJ-3 FLOOR-1) (TRACING T) (NOT (OCHK OBJ-3 F11-1))
 (NOT (FINDPOSS OBJ-3 FLOOR-1))      B34B33B51V16V17V10B1/

1342. B1-2   "DEF REF"
USING (REFERS OBJ-3 FLOOR-1) (CUROBJ OBJ-3 OBJ-2) (HASRELN OBJ-2 ON POS)
 (ERRREF OBJ-2 B7-1)
INSERTING (RELRESTRCHK OBJ-2 B7-1 ON FLOOR-1 POS) (CUROBJP OBJ-3 OBJ-2)
 (OLDREF OBJ-3)      B17/B13/

1343. B13-4  "REL RCHK EX"
USING (RELRESTRCHK OBJ-2 B7-1 ON FLOOR-1 POS) (FINDPOSS OBJ-2 BOX-2)
 (HASREL BOX-2 ON FLOOR-1 POS)
INSERTING (RELRESTRT OBJ-2 B7-1 ON FLOOR-1 POS)
 (NOT (RELRESTRCHK OBJ-2 B7-1 ON FLOOR-1 POS)) E33/

1344. E33-3  "TRACE R RESTR"
USING (RELRESTRT OBJ-2 B7-1 ON FLOOR-1 POS)

TRACING
RELRESTR OBJ-2 B7-1 ON FLOOR-1 POS


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (RELRESTR OBJ-2 B7-1 ON FLOOR-1 POS) (TRACING T)      F31/

1345. F31-4  "REL RESTR"
USING (RELRESTR OBJ-2 B7-1 ON FLOOR-1 POS) (FINDPOSS OBJ-2 BOX-1)
INSERTING (OCHK OBJ-2 B7-1) (NOT (FINDPOSS OBJ-2 BOX-1))       F11/F13/

1346. F13-6  "OBJ FND"
USING (OCHK OBJ-2 B7-1) (FINDPOSS OBJ-2 BOX-2)

TRACING
OBJ-2 REFERS BOX-2


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (REFERS OBJ-2 BOX-2) (TRACING T) (NOT (OCHK OBJ-2 B7-1))
 (NOT (FINDPOSS OBJ-2 BOX-2))  B16F23B1B1B15B53M2M5M1B3/

1347. B5-2   "DEF REF"
USING (REFERS OBJ-2 BOX-2) (CUROBJP OBJ-2 OBJ-1) (HASRELN OBJ-1 IN POS)
 (ERRREF OBJ-1 B4-1)
INSERTING (RELRESTRCHK OBJ-1 B4-1 IN BOX-2 POS) (OLDREF OBJ-2)  B18/B18/B18/B18
 B11B17/B13/

1348. B13-5  "REL RCHK EX"
USING (RELRESTRCHK OBJ-1 B4-1 IN BOX-2 POS) (FINDPOSS OBJ-1 BALL-3)
 (HASREL BALL-3 IN BOX-2 POS)
INSERTING (RELRESTRT OBJ-1 B4-1 IN BOX-2 POS)
 (NOT (RELRESTRCHK OBJ-1 B4-1 IN BOX-2 POS))

1349. B13-6  "REL RCHK EX"
USING (RELRESTRCHK OBJ-1 B4-1 IN BOX-2 POS) (FINDPOSS OBJ-1 BALL-4)
 (HASREL BALL-4 IN BOX-2 POS)
WARNING (OBJ-1 B4-1 IN BOX-2 POS) ALREADY UNDER RELRESTRT  *.
WARNING (OBJ-1 B4-1 IN BOX-2 POS) NOT UNDER RELRESTRCHK  *.
INSERTING (RELRESTRT OBJ-1 B4-1 IN BOX-2 POS)
 (NOT (RELRESTRCHK OBJ-1 B4-1 IN BOX-2 POS))  E33/

1350. E33-4  "TRACE R RESTR"
USING (RELRESTRT OBJ-1 B4-1 IN BOX-2 POS)

TRACING
RELRESTR OBJ-1 B4-1 IN BOX-2 POS


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (RELRESTR OBJ-1 B4-1 IN BOX-2 POS) (TRACING T)       F31/

1351. F31-5  "REL RESTR"
USING (RELRESTR OBJ-1 B4-1 IN BOX-2 POS) (FINDPOSS OBJ-1 BALL-1)
INSERTING (OCHK OBJ-1 B4-1) (NOT (FINDPOSS OBJ-1 BALL-1))

1352. F31-6  "REL RESTR"
USING (RELRESTR OBJ-1 B4-1 IN BOX-2 POS) (FINDPOSS OBJ-1 BALL-2)
WARNING (OBJ-1 B4-1) ALREADY UNDER OCHK  *.
INSERTING (OCHK OBJ-1 B4-1) (NOT (FINDPOSS OBJ-1 BALL-2))      F13/F11/F15/

1353. F15-9  "OBJ MULT"
USING (OCHK OBJ-1 B4-1) (FINDPOSS OBJ-1 BALL-3) (FINDPOSS OBJ-1 BALL-4)

TRACING
OBJ-1 AMBIG B4-1 BALL-3 BALL-4 _


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (TRACING T) (NOT (OCHK OBJ-1 B4-1))  B23B34B1/B51V16V17V10B28V32V37
 F29/B29B29V35V36V31V30B36B48B44B43M12M53M11M51B4B[illegible]
 G8E6N8AE4S1/

1354. B1-31  "SCAN ON"
```

```
USING (SCANFIN F11-1) (LEFTOF F11-1 T12-1)
INSERTING (SCAN T12-1) (SCANFIN T12-1) (NOT (SCANFIN F11-1))   T63/

1355. T63-3  "REL PRON"
USING (SCAN T12-1) (EQTHAT T12-1)
INSERTING (ISRELPRONW T12-1) (WORDEQ T12-1 THAT) (NOT (SCAN T12-1))
 (NOT (EQTHAT T12-1))  P1/

1356. P1-3   "RELPRON G"
USING (ISRELPRONW T17-1) (LEFTOF F11-1 T12-1) (ISNGLN F11-1 FLOOR)
INSERTING (ISRELPRON T12-1) (NOT (ISRELPRONW T12-1))   P8/N8AE6G9E481/

1357. S1-32  "SCAN ON"
USING (SCANFIN T12-1) (LEFTOF T12-1 I13-1)
INSERTING (SCAN I13-1) (SCANFIN I13-1) (NOT (SCANFIN T12-1))   T2T57G7G9T34T27
 T44T47T16T24T39T13T60T10T37T1/

1358. T1-9   "TAG COP"
USING (SCAN I13-1) (EQIS I13-1) (LEFTOF I13-1 R14-1)
INSERTING (ISCOP I13-1 POS) (WORDEQ I13-1 IS) (NOT (SCAN I13-1))
 (NOT (EQIS I13-1))   G10G32/

1359. G32-5  "COP +"
USING (ISCOP I13-1 POS)
WARNING (NEG) NOT UNDER COPSIGN  *.
INSERTING (COPSIGN POS) (NOT (COPSIGN NEG))   R11/N16N19G17G18/R1N8CA17G31/G5
 E6N8AE4S1/

1360. S1-33  "SCAN ON"
USING (SCANFIN I13-1) (LEFTOF I13-1 R14-1)
INSERTING (SCAN R14-1) (SCANFIN R14-1) (NOT (SCANFIN I13-1))   T410202106T31
 T53G1T60T7/

1361. T7-4   "TAG COLOR1"
USING (SCAN R14-1) (EQRED R14-1)
INSERTING (ISAVW R14-1 COLOR RED) (WORDEQ R14-1 RED) (NOT (SCAN R14-1))
 (NOT (EQRED R14-1))   A25/A19/A15/A16A17/

1362. A17-3  "AV G4"
USING (ISAVW R14-1 COLOR RED) (LEFTOF I13-1 R14-1) (ISCOP I13-1 POS)
INSERTING (ISPRED R14-1) (ISAV R14-1 COLOR RED POS)
 (NOT (ISAVW R14-1 COLOR RED)) R3P2F41/

1363. F41-3  "PRED RESTR"
USING (ISPRED R14-1) (CUROBJ OBJ-3 OBJ-2) (ISAV R14-1 COLOR RED POS)
INSERTING (PREDRESTRCHK OBJ-3 R14-1 COLOR RED POS) (OLDAV R14-1)      B27/B23
 /B29/B25/

1364. B25-1  "PRED RCHK RED"
USING (PREDRESTRCHK OBJ-3 R14-1 COLOR RED POS) (REFERS OBJ-3 FLOOR-1)
 (HASAV FLOOR-1 COLOR RED POS)
INSERTING (PREDREDUN:T OBJ-3 R14-1 COLOR RED POS)
 (NOT (PREDRESTRCHK OBJ-3 R14-1 COLOR RED POS))      E22/

1365. E22-1  "TRACE P RED"
USING (PREDREDUN:T ORJ-3 R14-1 COLOR RED POS)

TRACING
PREDREDUN OBJ-3 R14-1 COLOR RED POS


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (PREDREDUN ORJ-3 R14-1 COLOR RED POS) (TRACING T)    B41/

1366. B41-1  "BK PRED REDUN"
USING (PREDREDUN ORJ-3 R14-1 COLOR RED POS) (FINDPOSS OBJ-1 BALL-3)
 (HASAV BALL-3 COLOR RED POS)
INSERTING (FINDAMBIGP OBJ-3 R14-1 COLOR RED POS OBJ-3)  B43/B46/B46/B45/

1367. B45-1  "T AMB BK"
USING (FINDAMBIGP ORJ-3 R14-1 COLOR RED POS ORJ-3) (CUROBJP OBJ-3 OBJ-2)
INSERTING (FINDAMBIGP OBJ-2 R14-1 COLOR RED POS OBJ-3)
 (NOT (FINDAMBIGP OBJ-3 R14-1 COLOR RED POS OBJ-3))   B46/B44/B43/

1368. B43-1  "T AMB PRED"
USING (FINDAMBIGP ORJ-2 R14-1 COLOR RED POS OBJ-3) (CUROBJP OBJ-2 ORJ-1)
 (FINDPOSS OBJ-1 BALL-3) (HASAV BALL-3 COLOR RED POS) (CUROBJP OBJ-3 OBJ-2)
 (CUROBJ ORJ-3 OBJ-2)
INSERTING (PREDRESTR:T OBJ-1 R14-1 COLOR RED POS) (CUROBJ ORJ-3 OBJ-1)
 (CUROBJP OBJ-3 OBJ-1) (NOT (FINDAMBIGP ORJ-2 R14-1 COLOR RED POS OBJ-3))
 (NOT (CUROBJP OBJ-2 OBJ-1)) (NOT (CUROBJP OBJ-3 OBJ-2))
 (NOT (CUROBJ OBJ-3 OBJ-2)) (NOT (PREDREDUN OBJ-3 R14-1 COLOR RED POS))
 E23/

1369. E23-3  "TRACE P RESTR"
USING (PREDRESTR:T OBJ-1 R14-1 COLOR RED POS)

TRACING
PREDRESTR OBJ-1 R14-1 COLOR RED POS


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (PREDRESTR OBJ-1 R14-1 COLOR RED POS) (TRACING T)    F35/

1370. F35-3  "PRED RESTR"
USING (PREDRESTR OBJ-1 R14-1 COLOR RED POS) (FINDPOSS OBJ-1 BALL-4)
INSERTING (OCHK OBJ-1 R14-1) (NOT (FINDPOSS OBJ-1 BALL-4))     F11/F13/

1371. F13-7  "OBJ FND"
USING (OCHK OBJ-1 R14-1) (FINDPOSS OBJ-1 BALL-3)

TRACING
OBJ-1 REFERS BALL-3


WARNING (T) ALREADY UNDER TRACING  *.
INSERTING (REFERS OBJ-1 BALL-3) (TRACING T) (NOT (OCHK OBJ-1 R14-1))
 (NOT (FINDPOSS OBJ-1 BALL-3)) B34B33M1M5M2B53F23B3/B1B19B15B1/B51V19V17V10
 B28V32V37F29/B25B29V35V36V31V30V14B43N9AN33/A1/F53B38B48N9N01A9V12B1M4/S3F61/
 B24R12/R11/N1/F51V25B44B38B46M12M53M11M51B39B5B4B3B35M10V4BN21A1BE6G9E4S1/R/ :

1372. S4-3   "SCAN FIN"
USING (SCANFIN R14-1) (LEFTOF R14-1 RE-1) (ENDMARK RE-1) (SENTENCE S-1)
INSERTING (NPBOUND RE-1) (SENTBOUND S-1) (NOT (SCANFIN R14-1))  B53/

1373. B53-1  "NPBND UNDOP"
USING (NPBOUND RE-1) (CUROBJP OBJ-3 OBJ-1) (REFERS OBJ-3 FLOOR-1)
INSERTING (NOT (CUROBJP OBJ-3 OBJ-1))   B57B55/

1374. B55-3  "NPBND REDO"
USING (NPBOUND RE-1) (CUROBJP OBJ-1 MAIN)
INSERTING (CUROBJ OBJ-1 MAIN)   B33V14B43N9AN33/A1/F53V10V17/

1375. V17-5  "REPLY SQWR1"
USING (SENTBOUND S-1) (GSQWR S-1) (CUROBJ OBJ-1 MAIN) (REFERS OBJ-1 BALL-3)
 (HASREL BALL-3 IN BOX-2 POS)
INSERTING (QWRDESCR2 BALL-3) (DESCRIBE BALL-3) (DESCRIBE BOX-2)
 (QWRREPLY1 BALL-3 IN BOX-2 POS)      V18/

1376. V18-3  "REPLY SQWR1a"
USING (QWRDESCR2 BALL-3) (HASREL BALL-4 NEAR BALL-3 POS)
INSERTING (DESCRIBE BALL-4) (QWRREPLY2 BALL-3 BALL-4 NEAR POS)
 (NOT (QWRDESCR2 BALL-3))      D1/

1377. D1-9   "DESCRIBE"
USING (DESCRIBE BOX-2)
INSERTING (DESCRAV BOX-2 SIZE POS (THE)) (DESCRNX SIZE COLOR)
 (DESCRNX COLOR ISA) (NOT (DESCRIBE BOX-2))

1378. D1-10  "DESCRIBE"
USING (DESCRIBE BALL-3)
WARNING (SIZE COLOR) ALREADY UNDER DESCRNX  *.
WARNING (COLOR ISA) ALREADY UNDER DESCRNX  *.
INSERTING (DESCRAV BALL-3 SIZE POS (THE)) (DESCRNX SIZE COLOR)
 (DESCRNX COLOR ISA) (NOT (DESCRIBE BALL-3))

1379. D1-11  "DESCRIBE"
USING (DESCRIBE BALL-4)
WARNING (SIZE COLOR) ALREADY UNDER DESCRNX  *.
WARNING (COLOR ISA) ALREADY UNDER DESCRNX  *.
INSERTING (DESCRAV BALL-4 SIZE POS (THE)) (DESCRNX SIZE COLOR)
 (DESCRNX COLOR ISA) (NOT (DESCRIBE BALL-4))   D3/D12/D4/D11/

1380. D11-11  "DESCR AV POS"
USING (DESCRAV BALL-3 SIZE POS (THE)) (HASAV BALL-3 SIZE SMALL POS)
INSERTING (DESCRAV BALL-3 SIZE POS (THE SMALL))
 (DESCRIBED BALL-3 SIZE SMALL POS) (NOT (DESCRAV BALL-3 SIZE POS (THE)))

1381. D11-12  "DESCR AV POS"
USING (DESCRAV BALL-4 SIZE POS (THE)) (HASAV BALL-4 SIZE LARGE POS)
INSERTING (DESCRAV BALL-4 SIZE POS (THE LARGE))
 (DESCRIBED BALL-4 SIZE LARGE POS) (NOT (DESCRAV BALL-4 SIZE POS (THE)))
 D11/D3/D12/D4/D2/

1382. D2-17  "DESCR NEXT"
USING (DESCRAV BOX-2 SIZE POS (THE))
INSERTING (DESCRAV BOX-2 SIZE NEG (THE)) (NOT (DESCRAV BOX-2 SIZE POS (THE)))
```

```
1383. D2-18  "DESCR NEXT"
USING (DESCRAV BALL-3 SIZE POS (THE SMALL))
INSERTING (DESCRAV BALL-3 SIZE NEG (THE SMALL))
  (NOT (DESCRAV BALL-3 SIZE POS (THE SMALL)))

1384. D2-19  "DESCR NEXT"
USING (DESCRAV BALL-4 SIZE POS (THE LARGE))
INSERTING (DESCRAV BALL-4 SIZE NEG (THE LARGE))
  (NOT (DESCRAV BALL-4 SIZE POS (THE LARGE)))  D4/D12/D3/

1385. D3-9  "DESCR NEXT"
USING (DESCRAV BOX-2 SIZE NEG (THE)) (DESCRMX SIZE COLOR)
INSERTING (DESCRAV BOX-2 COLOR POS (THE)) (NOT (DESCRAV BOX-2 SIZE NEG (THE)))

1386. D3-10  "DESCR NEXT"
USING (DESCRAV BALL-3 SIZE NEG (THE SMALL)) (DESCRMX SIZE COLOR)
INSERTING (DESCRAV BALL-3 COLOR POS (THE SMALL))
  (NOT (DESCRAV BALL-3 SIZE NEG (THE SMALL)))

1387. D3-11  "DESCR NEXT"
USING (DESCRAV BALL-4 SIZE NEG (THE LARGE)) (DESCRMX SIZE COLOR)
INSERTING (DESCRAV BALL-4 COLOR POS (THE LARGE))
  (NOT (DESCRAV BALL-4 SIZE NEG (THE LARGE)))  D12/D4/D2/

1388. D2-20  "DESCR NEXT"
USING (DESCRAV BOX-2 COLOR POS (THE))
INSERTING (DESCRAV BOX-2 COLOR NEG (THE)) (NOT (DESCRAV BOX-2 COLOR POS (THE)))
     D3/D11/

1389. D11-13  "DESCR AV POS"
USING (DESCRAV BALL-3 COLOR POS (THE SMALL)) (HASAV BALL-3 COLOR RED POS)
INSERTING (DESCRAV BALL-3 COLOR POS (THE SMALL RED))
  (DESCRIBED BALL-3 COLOR RED POS) (NOT (DESCRAV BALL-3 COLOR POS (THE SMALL)))

1390. D11-14  "DESCR AV POS"
USING (DESCRAV BALL-4 COLOR POS (THE LARGE)) (HASAV BALL-4 COLOR GREEN POS)
INSERTING (DESCRAV BALL-4 COLOR POS (THE LARGE GREEN))
  (DESCRIBED BALL-4 COLOR GREEN POS)
  (NOT (DESCRAV BALL-4 COLOR POS (THE LARGE))) D11/D3/D2/

1391. D2-21  "DESCR NEXT"
USING (DESCRAV BALL-3 COLOR POS (THE SMALL RED))
INSERTING (DESCRAV BALL-3 COLOR NEG (THE SMALL RED))
  (NOT (DESCRAV BALL-3 COLOR POS (THE SMALL RED)))

1392. D2-22  "DESCR NEXT"
USING (DESCRAV BALL-4 COLOR POS (THE LARGE GREEN))
INSERTING (DESCRAV BALL-4 COLOR NEG (THE LARGE GREEN))
  (NOT (DESCRAV BALL-4 COLOR POS (THE LARGE GREEN)))  D2/D11/D3/D12/

1393. D12-3  "DESCR AV NEG"
USING (DESCRAV BOX-2 COLOR NEG (THE)) (HASAV BOX-2 COLOR RED NEG)
INSERTING (DESCRAV BOX-2 COLOR NEG (THE UN- RED))
  (DESCRIBED BOX-2 COLOR RED NEG) (NOT (DESCRAV BOX-2 COLOR NEG (THE))) D4/

1394. D4-9  "DESCR ISA"
USING (DESCRAV BALL-3 COLOR NEG (THE SMALL RED)) (DESCRMX COLOR ISA)
  (ISA BALL-3 BALL)
INSERTING (DESCRPHRASE BALL-3 (THE SMALL RED BALL))
  (NOT (DESCRAV BALL-3 COLOR NEG (THE SMALL RED)))

1395. D4-10  "DESCR ISA"
USING (DESCRAV BALL-4 COLOR NEG (THE LARGE GREEN)) (DESCRMX COLOR ISA)
  (ISA BALL-4 BALL)
INSERTING (DESCRPHRASE BALL-4 (THE LARGE GREEN BALL))
  (NOT (DESCRAV BALL-4 COLOR NEG (THE LARGE GREEN)))

1396. D4-11  "DESCR ISA"
USING (DESCRAV BOX-2 COLOR NEG (THE UN- RED)) (DESCRMX COLOR ISA)
  (ISA BOX-2 BOX)
INSERTING (DESCRPHRASE BOX-2 (THE UN- RED BOX))
  (NOT (DESCRAV BOX-2 COLOR NEG (THE UN- RED))) D25/

1397. D25-3  "DESCR REL+ INIT"
USING (QWRREPLY2 BALL-3 BALL-4 NEAR POS)
  (DESCRPHRASE BALL-4 (THE LARGE GREEN BALL))
INSERTING (QWRPHRASE2 BALL-4 (THE LARGE GREEN BALL) IS) D27/D28/

1398. D26-3  "DESCR REL+ POS"
USING (QWRPHRASE2 BALL-4 (THE LARGE GREEN BALL) IS)
  (QWRREPLY2 BALL-3 BALL-4 NEAR POS)
INSERTING (QWRPHRASE2 BALL-4 (THE LARGE GREEN BALL IS NEAR IT) AND)
  (NOT (QWRPHRASE2 BALL-4 (THE LARGE GREEN BALL) IS))
  (NOT (QWRREPLY2 BALL-3 BALL-4 NEAR POS))    D28/

1399. D28-3  "DESCR REL+ -"
USING (QWRPHRASE2 BALL-4 (THE LARGE GREEN BALL IS NEAR IT) AND)
INSERTING (REPLY (THE LARGE GREEN BALL IS NEAR IT))
  (NOT (QWRPHRASE2 BALL-4 (THE LARGE GREEN BALL IS NEAR IT) AND))     V8D22
  D21/

1400. D21-3  "DESCR REL INIT"
USING (QWRREPLY1 BALL-3 IN BOX-2 POS) (DESCRPHRASE BALL-3 (THE SMALL RED BALL))
INSERTING (QWRPHRASE1 BALL-3 (THE SMALL RED BALL) IS)  D22/

1401. D22-4  "DESCR REL POS"
USING (QWRPHRASE1 BALL-3 (THE SMALL RED BALL) IS)
  (QWRREPLY1 BALL-3 IN BOX-2 POS) (DESCRPHRASE BOX-2 (THE UN- RED BOX))
INSERTING (QWRPHRASE1 BALL-3 (THE SMALL RED BALL IS IN THE UN- RED BOX) AND)
  (NOT (QWRPHRASE1 BALL-3 (THE SMALL RED BALL) IS))
  (NOT (QWRREPLY1 BALL-3 IN BOX-2 POS)) D23D22D24/

1402. D24-3  "DESCR REL-"
USING (QWRPHRASE1 BALL-3 (THE SMALL RED BALL IS IN THE UN- RED BOX) AND)
INSERTING (REPLY (THE SMALL RED BALL IS IN THE UN- RED BOX))
  (NOT (QWRPHRASE1 BALL-3 (THE SMALL RED BALL IS IN THE UN- RED BOX) AND))
  V5V15D29V19/B3B84B81/B51/

1403. B51-1  "NPBND UNDO"
USING (NPBOUND RE-1) (CUROBJ OBJ-3 OBJ-1) (REFERS OBJ-3 FLOOR-1)
INSERTING (NOT (CUROBJ OBJ-3 OBJ-1))    B48B34N5N31A5V12B14M19F41/B24B12/R11/N1
  /F51V25V20V2V30V31V35V36V37V32V48V46V46V44V42V40

REPLY ((THE LARGE GREEN BALL IS NEAR IT))
  ((THE SMALL RED BALL IS IN THE UN- RED BOX))

ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BLOCK-1 BLOCK)
  (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-4 IN BOX-2 POS) (BALL-4 NEAR BALL-3 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)

AVRESTR (OBJ-3 R10-1 COLOR RED POS)
COPSIGN (POS)
CUROBJ (OBJ-1 MAIN)
CUROBJP (OBJ-1 MAIN)
DEFDET (T3-1) (T6-1) (T9-1)
DESCRIBED (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BOX-2 COLOR RED NEG)
DESCRMX (COLOR ISA) (SIZE COLOR)
DESCRPHRASE (BALL-3 (THE SMALL RED BALL)) (BALL-4 (THE LARGE GREEN BALL))
  (BOX-2 (THE UN- RED BOX))
DETSEEN (T3-1) (T6-1) (T9-1)
ENDMARK (LE-1) (RE-1)
ENTREF (OBJ-1 B4-1) (OBJ-2 B7-1) (OBJ-3 F11-1)
GSQWR (S-1)
GTYPED (S-1)
HASAV (BALL-1 COLOR BLUE POS) (BALL-2 COLOR BLUE POS) (BALL-2 SIZE SMALL POS)
  (BALL-3 SIZE SMALL POS) (BALL-3 COLOR RED POS) (BALL-4 SIZE LARGE POS)
  (BALL-4 COLOR GREEN POS) (BLOCK-1 SIZE LARGE POS) (BLOCK-1 COLOR GREEN POS)
  (BOX-2 COLOR RED NEG) (FLOOR-1 COLOR RED POS) (TABLE-1 COLOR RED POS)
HASREL (BALL-1 ON TABLE-1 POS) (BALL-1 NEAR BLOCK-1 POS) (BALL-2 ON BLOCK-1 POS)
  (BALL-3 IN BOX-2 POS) (BALL-4 IN BOX-2 POS) (BALL-4 NEAR BALL-3 POS)
  (BLOCK-1 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (BOX-2 ON FLOOR-1 POS)
  (BOX-2 ON TABLE-1 NEG)
HASRELN (OBJ-1 IN POS) (OBJ-2 ON POS)
ISA (BALL-1 BALL) (BALL-2 BALL) (BALL-3 BALL) (BALL-4 BALL) (BLOCK-1 BLOCK)
  (BOX-1 BOX) (BOX-2 BOX) (FLOOR-1 FLOOR) (TABLE-1 TABLE)
ISAV (R10-1 COLOR RED POS) (R14-1 COLOR RED POS)
ISCOP (I13-1 POS) (I2-1 POS)
ISDEF (OBJ-1) (OBJ-2) (OBJ-3)
ISNOUN (B4-1 BALL) (B7-1 BOX) (F11-1 FLOOR)
ISPRED (R14-1)
ISREL (I5-1 IN) (O8-1 ON)
ISRELPRON (T12-1)
LEFTOF (B4-1 I5-1) (B7-1 O8-1) (F11-1 T12-1) (I13-1 R14-1) (I2-1 T3-1)
  (I5-1 T6-1) (LE-1 W1-1) (O8-1 T9-1) (R10-1 F11-1) (R14-1 RE-1) (T12-1 I13-1)
  (T3-1 B4-1) (T6-1 B7-1) (T9-1 R10-1) (W1-1 I2-1)
NPBOUND (RE-1)
OLDAV (R10-1) (R14-1)
OLDREF (OBJ-2) (OBJ-3)
```

```
OLDREL (TS-1) (OB-1)
PREDREDUPAT (OBJ-3 R14-1 COLOR RED POS)
PREDRESTR (OBJ-1 R14-1 COLOR RED POS)
PREDRESTRT (OBJ-1 R14-1 COLOR RED POS)
REFERS (OBJ-1 BALL-3) (OBJ-2 BOX-2) (OBJ-3 FLOOR-1)
RELRESTR (OBJ-1 B4-1 IN BOX-2 POS) (OBJ-2 B7-1 ON FLOOR-1 POS)
RELRESTRT (OBJ-1 B4-1 IN BOX-2 POS) (OBJ-2 B7-1 ON FLOOR-1 POS)
REPLY ((THE LARGE GREEN BALL IS NEAR IT))
  ((THE SMALL RED BALL IS IN THE UN- RED BOX))
SENTBOUND (S-1)
SENTENCE (S-1)
TEST22 (T)
TEXT (22 (WHERE IS THE BALL IN THE BOX ON THE RED FLOOR THAT IS RED))
TRACING (T)
WORDEQ (B4-1 BALL) (B7-1 BOX) (F11-1 FLOOR) (I13-1 IS) (I2-1 IS) (I5-1 IN)
  (O8-1 ON) (R10-1 RED) (R14-1 RED) (T12-1 THAT) (T3-1 THE) (T6-1 THE)
  (T9-1 THE) (W1-1 WHERE)
```

Appendix E. AUGMENTATIONS TO MILIPS FOR WBLUE

```
BEGIN         % PS FOR MILISY -- MODIFIED FOR WBLUE %

EXPR MILIPS0: BEGIN NONFLUENT(LEFTOF);        % DIFFERENCES ONLY %

     % NO MILIPS P'S WERE DELETED. ONLY REPLACED AS SHOWN OR ADDED TO %

     % EXCEPT THAT THE X P'S WERE REPLACED AS A SET BY THE Y P'S %

B0: "SCAN LE" : SCANFIN(X) & ENDMARK(X) & LEFTOF(X,Y) & TEXT(N,Z)
   -> SCAN(Y) & SCANFIN(Y) & NEGATE(1) & NREPLY(0)
     & TRACING(TRACEPRINTM(N CONS '(INPUT TEXT IS Y') @ Z @ '(Y') ));

T31: "TAG REL1" : SCAN(X) & EQIN(X)
   -> ISRELW(X,'IN) & ISINDREL('IN) & WORDEQ(X,'IN) & NEGATE(ALL);
T34: "TAG REL2" : SCAN(X) & EQON(X)
   -> ISRELW(X,'ON) & ISINDREL('ON) & WORDEQ(X,'ON) & NEGATE(ALL);

T61: "TAG NOUN1" : SCAN(X) & EQPYRAMID(X)
   -> ISNOUNW(X,'PYRAMID) & WORDEQ(X,'PYRAMID) & NEGATE(ALL);

T66: "IT" : SCAN(X) & EQIT(X) & GRASPING(H,D)
   -> NPGONE(X) & EXISTS(OBJ) & CUROBJ(OBJ,'MAIN) & REFERS(OBJ,D)
     & CUROBJP(OBJ,'MAIN) & TRACING(TRACEPRINTM('OBJ,'REFERS,D'))
     & WORDEQ(X,'IT) & NEGATE(1) & ISNOUN(X,'IT) & ERRREF(OBJ,X);
T67: "IT?" : SCAN(X) & EQIT(X) & NOT( EXISTS(H,D) & GRASPING(H,D) )
   -> ERROR(X,'(NOT GRASPING)) & NOT SCANFIN(X) & NEGATE(1);

T71: "TAG UP" : SCAN(X) & EQUP(X) & EXPECTMOD(S,V) & SATISFIES(V,V EQ 'UP)
   -> WORDEQ(X,'UP) & ISIMPER(X) & NEGATE(ALL);
T72: "TAG DOWN" : SCAN(X) & EQDOWN(X) & EXPECTMOD(S,V) & SATISFIES(V,V EQ 'DOWN)
   -> WORDEQ(X,'DOWN) & IMPREL(S,'ON,'?) & ISIMPER(X) & NEGATE(ALL);

T81: "TO LEFT OF" : SCAN(X) & EQTO(X) & LEFTOF(X,Y) & EQTHE(Y) & LEFTOF(Y,Z)
     & EQLEFT(Z) & LEFTOF(Z,W) & EQOF(W)
   -> ISRELW(W,'TOLEFTOF) & ISCOMPREL('TOLEFTOF) & ISRELW(X,'TO) & OLDREL(X)
     & SCANFIN(W) & NOT SCANFIN(X) & NEGATE(1,2,A,B,B) & ISPRED(Z) & OLDAV(Z)
     & WORDEQ(X,'TO) & WORDEQ(Y,'THE) & WORDEQ(Z,'LEFT) & WORDEQ(W,'OF);
T82: "TO RIGHT OF" : SCAN(X) & EQTO(X) & LEFTOF(X,Y) & EQTHE(Y) & LEFTOF(Y,Z)
     & EQRIGHT(Z) & LEFTOF(Z,W) & EQOF(W)
   -> ISRELW(W,'TORIGHTOF) & ISCOMPREL('TORIGHTOF) & ISRELW(X,'TO) & OLDREL(X)
     & SCANFIN(W) & NOT SCANFIN(X) & NEGATE(1,2,A,B,B) & ISPRED(Z) & OLDAV(Z)
     & WORDEQ(X,'TO) & WORDEQ(Y,'THE) & WORDEQ(Z,'RIGHT) & WORDEQ(W,'OF);
T83: "IN FRONT OF" : SCAN(X) & EQIN(X) & LEFTOF(X,Y) & EQFRONT(Y) & LEFTOF(Y,Z)
     & EQOF(Z)
   -> ISRELW(Z,'INFRONTOF) & ISCOMPREL('INFRONTOF) & ISRELW(X,'TO) & OLDREL(X)
     & SCANFIN(Z) & NOT SCANFIN(X) & NEGATE(1,2,A,B) & ISPRED(Y) & OLDAV(Y)
     & WORDEQ(X,'IN) & WORDEQ(Y,'FRONT) & WORDEQ(Z,'OF);
T86: "BEHIND" : SCAN(X) & EQBEHIND(X)
   -> ISRELW(X,'BEHIND) & ISCOMPREL('BEHIND) & NEGATE(1,2) & WORDEQ(X,'BEHIND);
T87: "ABOVE" : SCAN(X) & EQABOVE(X)
   -> ISRELW(X,'ABOVE) & ISCOMPREL('ABOVE) & NEGATE(1,2) & WORDEQ(X,'ABOVE);
T88: "BELOW" : SCAN(X) & EQBELOW(X)
   -> ISRELW(X,'BELOW) & ISCOMPREL('BELOW) & NEGATE(1,2) & WORDEQ(X,'BELOW);

E6: ERRORS(X,EL) & LEFTOF(Y,X) & ENDMARK(Y) & WORDEQ(X,XW)
   -> REPLYO(XW CONS EL) & NEGATE(1);

END;

                    ......................

     % G = TOP-LEVEL GRAMMAR, A = ADJECTIVES %          % PAGE 2 %

EXPR MILGARP(): BEGIN

GS1: "A DEF I" : SCAN(X) & EQA(X) & SENTENCE(S) & GSI(S)
   -> DEFDET(X) & WORDEQ(X,'A) & IMPINDEF(X) & NEGATE(1,2);
G6: "A IND" : SCAN(X) & EQA(X) & SENTENCE(S) & GTYPED(S) & NOT GSQE(S)
     & NOT GSI(S)
   -> INDEFDET(X) & WORDEQ(X,'A) & NEGATE(1,2);

G41: "PICK INIT" : SCAN(X) & EQPICK(X) & SENTENCE(S) & NOT GTYPED(S)
   -> IMPTYPE(S,'PICK) & WORDEQ(X,'PICK) & EXPECTMOD(S,'UP) & GTYPED(S)
     & IMPREL(S,'IN,'HAND?-1) & ISIMPER(X) & GSI(S) & NEGATE(1,2);
G42: "STACK INIT" : SCAN(X) & EQSTACK(X) & SENTENCE(S) & NOT GTYPED(S)
   -> IMPTYPE(S,'STACK) & WORDEQ(X,'STACK) & EXPECTMOD(S,'UP) & GTYPED(S)
     & IMPREL(S,'ON,'?) & ISIMPER(X) & GSI(S) & NEGATE(1,2);
G43: "GRASP INIT" : SCAN(X) & EQGRASP(X) & SENTENCE(S) & NOT GTYPED(S)
```

```
    -> IMPTYPE(S,'GRASP) & WORDEQ(X,'GRASP) & GTYPED(S) & ISIMPER(X)
       & IMPREL(S,'IN,'HAND?-1) & GSI(S) & NEGATE(1,2);
   "PUT INIT" = SCAN(X) & EQPUT(X) & SENTENCE(S) & NOT GTYPED(S)
    -> IMPTYPE(S,'PUT) & WORDEQ(X,'PUT) & EXPECTMOD(S,'DOWN) & EXPECTMOD(S,'IN)
       & EXPECTMOD(S,'ON) & ISIMPER(X) & GTYPED(S) & GSI(S) & NEGATE(1,2);
G48: "AND" = SCAN(X) & EQAND(X) & GSI(S)
    -> NPBOUND(X) & NPBOUNDL(X) & CONJBOUND(S) & WORDEQ(X,'AND)
       & ISIMPER(X) & NEGATE(1,2);

END;
```

```
    % N - NOUN PHRASES AND NOUNS %                    % PAGE 3 %

EXPR MILN(); BEGIN

N2: "DEF DET" = DEFDET(X) & NOT( EXISTS(0,OP) & CUROBJ(0,OP) )
       & NOT DETSEEN(X)
    -> NPGCHK(X) & DETSEEN(X) & EXISTS(OBJ) & DEFFND(OBJ,X)
       & CUROBJP(OBJ,'MAIN) & CUROBJ(OBJ,'MAIN) & ISDEF(OBJ);
N3: "IMP INDEF" = DETSEEN(X) & IMPINDEF(X) & CUROBJ(O,P)
    -> IMPINDEF(O) & NEGATE(2);

N6: "INDEF DET" = INDEFDET(X) & NOT( EXISTS(0,OP) & CUROBJ(0,OP) )
       & NOT DETSEEN(X)
    -> NPGCHK(X) & DETSEEN(X) & EXISTS(OBJ) & CUROBJ(OBJ,'MAIN) & ISINDEF(OBJ);

N9: "NP GRAM 1" = NPGCHK(X) -> NPGCHK1(X) & NPGCHK2(X);
N9A: "NP GRAM" = NPGCHK1(X) & LEFTOF(W,X) & WORDEQ(W,WW)
       & SATISFIES(WW,WW EQ 'THERE) & GSQE(S) & CUROBJ(O,P) & ISDEF(O)
    -> NEGATE(1);
N9B: "NP GRAM" = NPGCHK1(X) & LEFTOF(W,X) & ISREL(W,WW) -> NEGATE(1);
N9C: "NP GRAM" = NPGCHK1(X) & LEFTOF(W,X) & ISCOP(W,1) -> NEGATE(1);
N9D: "NP GRAM" = NPGCHK1(X) & LEFTOF(W,X) & ENDMARK(W) -> NEGATE(1);
N9E: "NP GRAM" = NPGCHK1(X) & LEFTOF(W,X) & ISIMPER(W) -> NEGATE(1);
N10: "NP UNGRAM CHK" = NPGCHK2(X) -> NPGCHK3(X) & NEGATE(1);
N10U: "NP UNGRAM" = NPGCHK1(X) & NPGCHK3(X)
    -> ERROR(X,'(GRAMMAR)) & NEGATE(ALL);

N41: "ISA PYRAMID" = MAKISA(X,XW,O,P) & SATISFIES(XW,XW EQ 'PYRAMID)
    -> EXISTS(PYRAMID) & ADDAV(PYRAMID,O,O) & ISA(PYRAMID,'PYRAMID)
       & CUROBJ(PYRAMID,P) & REFERS(PYRAMID,PYRAMID) & ERREF(PYRAMID,X)
       & NEWOBJ(PYRAMID) & NEGATE(1);

END;
```

```
    % F - FIND REFERENTS %                    % PAGE 4 %

EXPR MILF(); BEGIN

F23: "N INCON" = NRESTR(O,X,XW) & REFERS(O,OA) & NOT ISA(OA,XW)
    -> NULLREF(O,X) & NEGATE(ALL);      % INCLUDED BECAUSE BUG IN THE ISA %

F31: "REL RESTR" = RELRESTR2(O,X,R,O7,S) & NOT ISINDREL(R) & NOT ISCOMPREL(R)
       & FINDPOSS(O,O3) & NOT HASREL(O3,R,O7,S)
    -> OCHK(O,X) & NEGATE(4);
F32: "REL RESTR IND" = RELRESTR2(O,X,R,O2,S) & ISINDREL(R) & FINDPOSS(O,O3)
       & NOT HASINDREL(O3,R,O7)
    -> OCHK(O,X) & NEGATE(3);
F32C: "REL RESTR COMP" = RELRESTR2(O,X,R,O7,S) & ISCOMPREL(R) & FINDPOSS(O,O3)
       & NOT HASINDREL(O3,R,O2)
    -> OCHK(O,X) & NEGATE(3);
F33: "REL RESTR1?" = RELRESTR(O,X,R,O7,S)
    -> RELRESTR1(O,X,R,O7,S) & RELRESTR2(O,X,R,O7,S);
F34: "SAVE RESTR" = RELRESTR1(O,X,R,O7,S) & GSI(SN) & CUROBJP(O,P)
       & SATISFIES(P,P EQ 'MAIN) & EXPECTMOD(SN,R) & FINDPOSS(O,O3)
       & VNEQ(O3,O7) & FINDPOSS(O,O4) & VNEQ(O4,O7) & VNEQ(O3,O4)
       & NOT( EXISTS(O5) & FINDPOSS(O,O5) & VNEQ(O5,O3) & VNEQ(O5,O4)
            & VNEQ(O5,O7) )
       & HASREL(O3,R,O7,S) & NOT HASREL(O4,R,O7,S) & NOT HASINDREL(O4,R,O7)
       % ASSUMES NEG NOT USED %
       % ONLY SAVES A POSSIBLE ALTERNATIVE WHEN THAT IS UNIQUE %
    -> IMPRESTR(O,O4,R,O2) & NEGATE(1);
F341: "SAVE RESTR 1" = RELRESTR1(O,X,R,O2,S) & GSI(SN) & CUROBJP(O,P)
       & SATISFIES(P,P EQ 'MAIN) & EXPECTMOD(SN,R) & FINDPOSS(O,O3)
       & VNEQ(O3,O7) & FINDPOSS(O,O4) & VNEQ(O4,O7) & VNEQ(O3,O4)
       & NOT( EXISTS(O5) & FINDPOSS(O,O5) & VNEQ(O5,O3) & VNEQ(O5,O4)
            & VNEQ(O5,O2) )
       & HASINDREL(O3,R,O7) & NOT HASREL(O4,R,O2,S) & NOT HASINDREL(O4,R,O2)
       % ASSUMES NEG NOT USED %
    -> IMPRESTR(O,O4,R,O2) & NEGATE(1);
       % HERE, NEED TWO MORE F'S TO HANDLE CASE WHERE HAVE
            IMPINDEF AS THE FINDPOSS'S - WANT TO CHOOSE ONE TO
            BE THE IMPRESTR %

F61: "COMP LEFT" = CONVIND(R,O2) & SATISFIES(R,R EQ 'TOLEFTOF) & FINDPOSS(O,O3)
       & LOCAT(O3,X1,Y1,Z1) & LOCAT(O2,X2,Y2,Z2)
       & SATISFIES2(X1,X2,X1 ?=LESS X2)
    -> HASINDREL(O3,R,O2) & NEGATE(1);
F62: "COMP RIGHT" = CONVIND(R,O2) & SATISFIES(R,R EQ 'TORIGHTOF)
       & FINDPOSS(O,O3) & LOCAT(O3,X1,Y1,Z1) & LOCAT(O2,X2,Y2,Z2)
       & SATISFIES2(X1,X2,X1 ?=GREAT X2)
    -> HASINDREL(O3,R,O2) & NEGATE(1);
F63: "COMP FRONT" = CONVIND(R,O2) & SATISFIES(R,R EQ 'INFRONTOF)
       & FINDPOSS(O,O3) & LOCAT(O3,X1,Y1,Z1) & LOCAT(O2,X2,Y2,Z2)
       & SATISFIES2(Y1,Y2,Y1 ?=LESS Y2)
    -> HASINDREL(O3,R,O2) & NEGATE(1);
F64: "COMP BEHIND" = CONVIND(R,O2) & SATISFIES(R,R EQ 'BEHIND) & FINDPOSS(O,O3)
       & LOCAT(O3,X1,Y1,Z1) & LOCAT(O2,X2,Y2,Z2)
       & SATISFIES2(Y1,Y2,Y1 ?=GREAT Y2)
    -> HASINDREL(O3,R,O2) & NEGATE(1);
F65: "COMP ABOVE" = CONVIND(R,O2) & SATISFIES(R,R EQ 'ABOVE) & FINDPOSS(O,O3)
       & LOCAT(O3,X1,Y1,Z1) & LOCAT(O2,X2,Y2,Z2)
       & SATISFIES2(Z1,Z2,Z1 ?=GREAT Z2)
    -> HASINDREL(O3,R,O2) & NEGATE(1);
F66: "COMP BELOW" = CONVIND(R,O2) & SATISFIES(R,R EQ 'BELOW) & FINDPOSS(O,O3)
       & LOCAT(O3,X1,Y1,Z1) & LOCAT(O2,X2,Y2,Z2)
       & SATISFIES2(Z1,Z2,Z1 ?=LESS Z2)
    -> HASINDREL(O3,R,O2) & NEGATE(1);

END;
```

```
    % B - BACKUP REFERENTS %                    % PAGE 5 %

EXPR MILB(); BEGIN

B10: "REL RESTR IND-" = RELRESTRCHK(O,X,R,O7,S) & NOT ISINDREL(R)
       & NOT ISCOMPREL(R)
    -> RELRESTRCHK2(O,X,R,O7,S) & NEGATE(1);
B10C: "REL RESTR COMP" = RELRESTRCHK(O,X,R,O2,S) & ISCOMPREL(R)
    -> CONVIND(R,O2) & RELRESTRCHK2(O,X,R,O2,S) & NEGATE(1);    % SEE F60'S %
B10I: "REL RESTR IND" = RELRESTRCHK(O,X,R,O2,S) & ISINDREL(R)
    -> CHAINREL(R,O2,R,O2) & RELRESTRCHK2(O,X,R,O2,S) & NEGATE(1);
B10J: "REL CHAIN IN" = CHAINREL(R,O,R,O) & SATISFIES(R,R EQ 'IN)
       & HASREL(O2,R,O,S) & SATISFIES(S,S EQ 'POS)
    -> CHAINREL(R,O,'ON,O2) & HASINDREL(O2,R,O) & NEGATE(1);
B10K: "REL CHAIN IN-" = CHAINREL(R,O,R2,O2) & SATISFIES(R2,R2 NEQ 'IN)
       & ISA(O,W) & NOT SATISFIES(W,W EQ 'TABLE)
       & HASREL(O3,R2,O2,S) & SATISFIES(S,S EQ 'POS)
    -> CHAINREL(R,O,R2,O3) & HASINDREL(O3,R,O) & NEGATE(1);
B10L: "REL CHAIN TABLE" = CHAINREL(R,O,R2,O2) & ISA(O,W)
       & SATISFIES(W,W EQ 'TABLE) & HASREL(O3,R,O,S) & SATISFIES(S,S EQ 'POS)
    -> HASINDREL(O3,R,O) & NEGATE(1);

B11: "REL RCHK NEW" = RELRESTRCHK2(O,X,R,O2,S) & NEWOBJ(O)
       & NOT HASREL(O,R,O7,S)
       & NOT( EXISTS(N) & HASREL(O,R,O2,N) & VNEQ(N,S) )
    -> HASREL(O,R,O7,S) & NEGATE(1);
B13: "REL RCHK EX" = RELRESTRCHK2(O,X,R,O2,S) & FINDPOSS(O,OA)
       & HASREL(OA,R,O7,S)
    -> RELRESTRT(O,X,R,O7,S) & NEGATE(1);
B131: "REL RCHK EX" = RELRESTRCHK2(O,X,R,O2,S) & FINDPOSS(O,OA)
       & HASINDREL(OA,R,O2)
    -> RELRESTRT(O,X,R,O2,S) & NEGATE(1);
B14: "REL RCHK QW" = RELRESTRCHK2(O,X,R,O2,S) & GSQW(SN) & CUROBJ(O,P)
       & SATISFIES(P,P EQ 'MAIN)
    -> RELRESTRT(O,X,R,O2,S) & NEGATE(1);
B15: "REL RCHK RED" = RELRESTRCHK2(O,X,R,O7,S) & REFERS(O,OA)
       & HASREL(OA,R,O2,S)
    -> RELREDUNT(O,X,R,O7,S) & NEGATE(1);
B151: "REL RCHK RED" = RELRESTRCHK2(O,X,R,O2,S) & REFERS(O,OA)
       & HASINDREL(OA,R,O7)
    -> RELREDUNT(O,X,R,O7,S) & NEGATE(1);
B17: "REL RCHK ERR" = RELRESTRCHK2(O,X,R,O7,S) & FINDPOSS(O,OX)
       & NOT( EXISTS(OA) & FINDPOSS(O,OA) & HASREL(OA,R,O7,S) )
```

```
      & NOT( EXISTS(OA) & FINDPOSS(O,OA) & HASINDREL(OA,R,D2) )
      & SENTENCE(SN) & NOT( EXISTS(P) & GSQW(SN) & CUROBJ(O,P)
           & SATISFIES(P,P EQ 'MAIN) )
      & NOT( EXISTS(P) & GSI(SN) & CUROBJP(O,P) & IMPINDEF(O) )
   -> ERROR(X,'(WHICH ONE ??)) & NEGATE(1);
B16: "REL RCHK INC" = RELRESTRCHK7(O,X,R,D2,S) & NOT NEWOBJ(O) & REFERS(O,OA)
      & NOT HASREL(OA,R,D2,S) & NOT HASINDREL(OA,R,D2)
   -> RELINCONT(O,X,R,D2,S) & NEGATE(1);
B16: "REL RCHK INC-" = RELRESTRCHK7(O,X,R,D2,S) & REFERS(O,OA)
   •  & HASREL(OA,R,D2,N) & VNEQ(N,S)
   -> RELINCONT(O,X,R,D2,S) & NEGATE(1);
B16C: "REL RCHK CHOICE" = RELRESTRCHK7(O,X,R,D2,S) & FINDPOSS(O,OX)
      & IMPINDEF(O)
      & NOT( EXISTS(OX2) & FINDPOSS(O,OX2) & HASREL(OX2,R,D2,S) )
      & NOT( EXISTS(OX2) & FINDPOSS(O,OX2) & HASINDREL(OX2,R,D2) )
      % NOT SURE WHETHER THIS IS ENOUGH: IN CASE B13 OR 131, MAY STILL
           WANT TO CHOOSE AND THAT CHOICE MAY NOT BE MADE UNTIL NPBOUND,
           MAY BE TOO LATE TO UNRAVEL OR PROPOGATE %
   -> IMPCHOOSE(O) & RELRESTRCHK7(O,X,R,D2,S);
B19I: "REL RCHK IMP" = RELRESTRCHK7(O,X,R,D2,S) & GSI(SN) & FINDPOSS(O,D2)
      & CUROBJP(O,P) & SATISFIES(P,P EQ 'MAIN)
   -> OCHK(O,X) & RELRESTRCHK2(O,X,R,D2,S) & NEGATE(3);

B311: "BK REL REDUN" = RELREDUN(O,X,R,D2,S) & FINDPOSS(O3,D4)
      & HASINDREL(O4,R,D7)
   -> FINDAMBIGR(O,X,R,D2,S,O);

B331: "F AMB REL" = FINDAMBIGR(O,X,R,D2,S,1) & CURORJP(O,P) & FINDPOSS(P,OA)
      & HASINDREL(OA,R,D7) & REFERS(O4,D7) & CURORJP(O4,O3) & CURORJ(O4,O3)
   -> RELRESTRIT(P,X,R,D7,S) & CURORJ(O4,P) & CURORJP(O4,P) & NEGATE(1,2,8,7)
      & NOT RELREDUN(1,X,R,D2,S);

B36: "F AMB -" = FINDAMBIGR(O,X,R,D2,S,1) & CUROBJP(O,P) & FINDPOSS(P,OA)
      & NOT HASREL(OA,R,D2,S) & NOT HASINDREL(OA,R,D7)
      & NOT( EXISTS(O3) & FINDPOSS(P,O3) & HASREL(O3,R,D7,S) )
      & NOT( EXISTS(O3) & FINDPOSS(P,O3) & HASINDREL(O3,R,D2) )
   -> NEGATE(1);

B55: "NPBND REDO"= NPBOUND(X)
      & NOT( EXISTS(O,OX) & FINDPOSS(O,OX)
           & NOT( EXISTS(SN,P) & GSQW(SN) & CUROBJP(O,P)
                & SATISFIES(P,P EQ 'MAIN) ) )
      & CUROBJP(O,P) & SATISFIES(P,P EQ 'MAIN) & NOT CUROBJ(O,P)
   -> CUROBJ(O,P);
   % GSQW CASE IS V18 %
B56: "NPBND REDO" = NPBOUND(X) & FINDPOSS(O,OX) & CURORJP(O,P) & IMPINDEF(O)
      & NOT( EXISTS(O2,OX2) & FINDPOSS(O2,OX2) & VNEQ(O2,O) & IMPINDEF(O2)
           & SATISFIES2(O2,O,O LEXORDER O7) )
      & NOT CURORJ(O,P)  % LEXORDER BAD IF > 10 OBJ'S (LONG SENT.) %
   -> CUROBJ(O,P);
B57: "NPBND ERR" = NPBOUND(X) & FINDPOSS(O,OX) & ERRREF(O,L)
      & NOT( EXISTS(SN,P) & GSQW(SN) & CUROBJP(O,P)
           & SATISFIES(P,P EQ 'MAIN) )
      & NOT( EXISTS(O7) & IMPINDEF(O7) )
   -> ERROR(L,'(WHICH ONE ??));
B58: "NPBND CHOICE" = NPBOUND(X) & FINDPOSS(O,OX) & GSI(S) & CUROBJ(O,P)
      & IMPINDEF(O)
   -> IMPCHOOSE(O) & NPBOUND(X);
B58C: "CHOOSE" = IMPCHOOSE(O) & FINDPOSS(O,OX) & NOT IMPCHOICE(OX)
      & NOT( EXISTS(OX7) & FINDPOSS(O,OX7) & NOT IMPCHOICE(OX2)
           & SATISFIES7(OX,OX7,NOT(OX7 LEXORDER OX)) )
   -> ERSFINDPOSS(O) & TRACING(TRACEPRINTM('CHOOSING,OX,'FOR,O))
      & REFERS(O,OX) & IMPCHOICE(OX) & NEGATE(1);
B58E: "ERS POSS" = ERSFINDPOSS(O) & FINDPOSS(O,P) -> NEGATE(ALL);
B58F: "CHOOSE?" = IMPCHOOSE(O)
      & NOT( EXISTS(OX) & FINDPOSS(O,OX) & NOT IMPCHOICE(OX) )
      & ERRREF(O,X)
   -> ERROR(X,'(NO MORE CHOICES)) & NEGATE(1);
B59: "NPBND DEL" = NPBOUNDL(W) & NOT( EXISTS(S) & GSI(S) & CONJBOUND(S) )
   -> NOT NPBOUND(W) & NEGATE(1);
B59I: "NPBND DEL IMP" = NPBOUNDL(W) & GSI(S) & CONJBOUND(S) & CUROBJP(O,P)
      & SATISFIES(P,P EQ 'MAIN) & REFERS(O,OX)
   -> IMPOBJ(S,OX) & NOT NPBOUND(W) & NEGATE(1,3,4) & NOT CUROBJ(O,P);

END;
```

....

```
      % M - SEMANTIC CASES FOR DIFFERENT SENTENCE TYPES %    % PAGE 8 %

EXPR MIL(M); BEGIN
```

```
M8: "PRED REDUN SD" = PREDREDUN(O1,X,A,V,S) & GSD(SN) & CUROBJP(O1,P)
      % INCLUDED AS DIFFERENCE BECAUSE BUG IN FIRST ARG OF PREDREDUN %
      & SATISFIES(P,P EQ 'MAIN) & REFERS(O1,OA)
      & NOT( EXISTS(O2,O3,D4) & CUROBJP(O2,O3) & FINDPOSS(O2,D4) )
   -> HASAV(OA,A,V,S) & NEGATE(1);

M11: "ANSREL I" = RELINCON(O,X,R,D2,S) & CUROBJP(O,P) & SATISFIES(P,P EQ 'MAIN)
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSE(SN) & NOT GSQW(SN)
      & NOT GSQWR(SN) & NOT GSI(SN)          % THAT LEAVES GSQE OR GSQD %
   -> ANSRELINC(O,X,R,D2,S) & NEGATE(1);
M12: "ANSREL R" = RELREDUN(O,X,R,D2,S) & CUROBJP(O3,P)
      & SATISFIES(P,P EQ 'MAIN)
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSE(SN) & NOT GSQW(SN)
      & NOT GSQWR(SN) & NOT GSI(SN)          % LEAVING GSQE OR GSQD %
      & NOT( EXISTS(O4,D5) & FINDPOSS(O4,D5) & HASREL(D5,R,D2,S) )
      % JUST IN CASE RELRED IS THE Q BEING ASKED; ANS WILL BE NO %
   -> ANSRELRED(O3,R,D2,S) & NEGATE(1);
M13: "ANSPRED I" = PREDINCON(O,X,A,V,S) & CUROBJP(O,P)
      & SATISFIES(P,P EQ 'MAIN) & GSQE(SN)
   -> ANSPRED(O,A,V,S) & NEGATE(1);

M51: "REL INCON ERR" = RELINCON(O,X,R,D2,S) & CUROBJP(O,P)
      & SATISFIES(P,P EQ 'MAIN)       % THIS IS FOR GSE, GSQW, GSQWR %
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSQE(SN) & NOT GSQD(SN) & NOT GSI(SN)
   -> ERROR(X,'(INCONSISTENT) );
M53: "PRED INCON ERR" = PREDINCON(O,X,A,V,S) & CUROBJP(O,P)
      & SATISFIES(P,P EQ 'MAIN)       % THIS IS FOR GSE, GSQW, GSQWR, GSI %
      & SENTENCE(SN) & NOT GSD(SN) & NOT GSQE(SN) & NOT GSQD(SN)
   -> ERROR(X,'(INCONSISTENT) );

M61: "REL INCON IMP" = RELINCON(O,X,R,D2,S) & GSI(SN) & CUROBJP(O,O3)
      & SATISFIES(O3,O3 EQ 'MAIN) & EXPECTMOD(SN,R)
      & NOT( EXISTS(R2,D4) & IMPREL(SN,R2,D4) )
   -> IMPREL(SN,R,D2) & NEGATE(1);
M62: "REL INCON IMP-" = RELINCON(O,X,R,D2,S) & GSI(SN) & CUROBJP(O,O3)
      & SATISFIES(O3,O3 EQ 'MAIN) & EXPECTMOD(SN,R) & IMPREL(SN,R2,D4)
   -> ERROR(X,'(INCONSISTENT));
M63: "IMP REV" = SENTBOUND(SN) & GSI(SN)
      & NOT( EXISTS(R,D,D4) & IMPREL(SN,R,D) )
      & IMPRESTR(O1,D2,R1,O3) & REFERS(O1,OX)
      % IN CASE OF COMPOUND OBJECT OF THE IMPERATIVE, ALL THIS APPLIES
           ONLY TO THE LAST, SINCE ASSUMING "AND" FINISHES OFF
           THE OTHER OBJECTS %
   -> SENTBOUND(SN) & REFERS(O1,D7) & IMPREL(SN,R1,D3)
      & TRACING(TRACEPRINTM('BACKUP,D1,'REFERS,D2)) & NEGATE(5);
M64: "IMP SEL REDUN" = SENTBOUND(SN) & GSI(SN)
      & NOT( EXISTS(R,D1) & IMPREL(SN,R,D1) )
      & NOT( EXISTS(R,D1,D7,D3) & IMPRESTR(O1,D2,R,D3) )
      & RELREDUN(O,X,R1,D4,S) & EXPECTMOD(SN,R1)
      & NOT( EXISTS(O5,X2,R2,D6,S2) & RELREDUN(O5,X2,R2,D6,S2)
           & EXPECTMOD(SN,R2) & VNEQ(O4,D6) )
   -> IMPREL(SN,R1,D4);
M64A: "IMP SEL REDUN-" = SENTBOUND(SN) & GSI(SN)
      & NOT( EXISTS(R,D1) & IMPREL(SN,R,D1) )
      & NOT( EXISTS(R,D1,D7,D3) & IMPRESTR(O1,D2,R,D3) )
      & RELREDUN(O,X,R1,D4,S) & EXPECTMOD(SN,R1) & RELREDUN(O5,X2,R2,D6,S2)
      & EXPECTMOD(SN,R2) & VNEQ(O4,O6) & SATISFIES2(O4,D6,D6 LEXORDER O4)
   -> ERROR(X,'(AMBIG IMPER REL)); % MAY FIRE MULTI IF GT 2 %
M65: "IMP REDUN" = SENTBOUND(SN) & GSI(SN)
      & NOT( EXISTS(R,D1) & IMPREL(SN,R,D1) )
      & NOT( EXISTS(R,D1,D2,D3) & IMPRESTR(O1,D2,R,D3) )
      & NOT( EXISTS(R,D1,Y,D7,S) & RELREDUN(O1,Y,R,D2,S) & EXPECTMOD(SN,R) )
      & ENDMARK(X) & NOT( EXISTS(V) & LEFTOF(X,V) )
   -> ERROR(X,'(REDUNDANT COMMAND) );
M66: "IMP REDUN GRASP" = SENTBOUND(SN) & GSI(SN) & IMPREL(SN,R,D1)
      & SATISFIES(R,R EQ 'IN) & SATISFIES(O1,O1 EQ 'HAND7-1)
      & GRASPING(O1,D7) & IMPOBJ(SN,D7) & ENDMARK(X)
      & NOT( EXISTS(V) & LEFTOF(X,V) )
   -> ERROR(X,'(REDUNDANT COMMAND));

M71: "IMP OBJ" = SENTBOUND(S) & GSI(S) & IMPREL(S,R,D) & CUROBJ(O1,P)
      & SATISFIES(P,P EQ 'MAIN) & REFERS(O1,OA) & NOT IMPOBJ(S,OA)
   -> IMPOBJ(S,OA);

      % M80'S: DISPATCH TO W'S AND Q'S ACCORDING TO IMPTYPE, OBJ, REL %

M81: "CMD PICKUP" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PICK)
      & NOT( EXISTS(W) & EXPECTMOD(SN,W) ) & IMPOBJ(SN,D)
   -> WBPINIT('GT) & PICKUP('GT,D) & CHECKPICKUP(D);
M81F: "CMD PICK ?" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PICKUP)
      & EXPECTMOD(SN,W)
   -> SENTBOUND(SN) & REPLYO('(UP ??)) & NEGATE(4);
M82: "CMD PUTON" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PUT)
```

```
      & IMPOBJ(SN,O) & IMPREL(SN,R,O2) & SATISFIES(R,R EQ 'ON)
      & NOT SATISFIES(O2,O2 EQ '??) & ISA(O2,W)
      & NOT SATISFIES(W,W EQ 'PYRAMID)
   -> WBPINIT('GT) & PUTON('GT,O,O2) & CHECKPUTON(O,R,O2);
      % MAY FIRE MULTIPLY %
MB2P; "CMD PUTON PYR" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PUT)
      & IMPOBJ(SN,O) & IMPREL(SN,R,O2) & SATISFIES(R,R EQ 'ON)
      & ISA(O2,W) & SATISFIES(W,W EQ 'PYRAMID)
   -> REPLYO('(CANT PUT ON PYRAMID));
MB3; "CMD PUTIN" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PUT)
      & IMPOBJ(SN,O) & IMPREL(SN,R,O2) & SATISFIES(R,R EQ 'IN)
      & ISA(O2,W) & SATISFIES(W,W EQ 'BOX)
   -> TRACEPUTIN('T) & WBPINIT('GT) & PUTON('GT,O,O2)
      & CHECKPUTON(O,R,O2); % MAY FIRE MULTIPLY %
MB3P; "CMD PUTIN -BOX" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PUT)
      & IMPOBJ(SN,O) & IMPREL(SN,R,O2) & SATISFIES(R,R EQ 'IN)
      & ISA(O2,W) & NOT SATISFIES(W,W EQ 'BOX)
   -> REPLYO('(CAN ONLY PUT IN BOX));
MB3T; "TRACE PUTIN" = TRACEPUTIN(X)
   -> TRACING(TRACEPRINTM(<'PUTIN,'STARTS,'WITH,'PUTON>)) & NEGATE(1);
MB4; "CMD PUT DOWN" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PUT)
      & IMPOBJ(SN,O) & IMPREL(SN,R,O2) & SATISFIES(R,R EQ 'ON)
      & SATISFIES(O2,O2 EQ '??) & GRASPING(H,O) & LOCAT(O,X,Y,Z)
   -> WBPINIT('GT) & PUTDOWN('GT,O) & CHECKPUTDOWN(O,X,Y,Z);
MB4F; "CMD PUT DOWN ?" = SENTBOUND(SN) & IMPTYPE(SN,V)
      & SATISFIES(V,V EQ 'PUT)
      & IMPOBJ(SN,O) & IMPREL(SN,R,O2) & SATISFIES(R,R EQ 'ON)
      & SATISFIES(O2,O2 EQ '??) & NOT( EXISTS(H) & GRASPING(H,O) )
   -> REPLYO('(NOT GRASPING) @ O);
MB5; "CMD PUT ?" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'PUT)
      & IMPOBJ(SN,O) & NOT( EXISTS(R,O2) & IMPREL(SN,R,O2) )
   -> REPLYO('(PUT WHERE ??) );
MB6; "CMD STACKUP" = SENTBOUND(SN) & IMPTYPE(SN,V) & SATISFIES(V,V EQ 'STACK)
      & NOT( EXISTS(W) & EXPECTMOD(SN,W) ) & IMPOBJ(SN,O)
   -> WBPINIT('GT) & STACKUP('GT,O) & CHECKSTACKUP(O);
MB6F; "CMD STACK ?" = SENTBOUND(SN) & IMPTYPE(SN,V)
      & SATISFIES(V,V EQ 'STACKUP) & EXPECTMOD(SN,W)
   -> SENTBOUND(SN) & REPLYO('(UP ??)) & NEGATE(4);
MB7; "IMP NO OBJ" = SENTBOUND(SN) & IMPTYPE(SN,V)
      & NOT( EXISTS(O) & IMPOBJ(SN,O) )
      & NOT( EXISTS(O,P,DA) & CUROBJ(O,P) & SATISFIES(P,P EQ 'MAIN)
            & REFERS(O,DA) )
   -> REPLYO(V CONS '(WHAT ??));
MB9; "WB P INIT" = WBPINIT(GT) & SENTBOUND(SN)
   -> EVENTTIME(O) & CHOICECOUNT(O) & HASLEVEL(GT,O) & NEGATE(ALL);
      % SENTBOUND STUFF PREVENTS EFFECTS OF CHANGES IN SCENE, EG ON MB6 %

END;
```

- -

```
      % V = REPLY, D = DESCRIBE %                    % PAGE 7 %

EXPR MILVO(); BEGIN

VO; "COUNT REPLY" = REPLYO(R) & NREPLY(N)
   -> REPLY(N+1,R) & NREPLY(N+1) & NEGATE(ALL);
V2; "REPLY SO" = SENTBOUND(S) & GSO(S) -> REPLYO('(OKAY));
V5; "REPLY QUIT" = REPLY(N,R) & SCANFIN(X) -> NEGATE(2);

V18; "REPLY SQWP" = QWREPLY(X) & DESCRPHRASE(X,L) & NREPLY(N)
      & NOT( EXISTS(Y,M) & QWREPLY(Y) & VNEQ(Y,X) & DESCRPHRASE(Y,M)
            & SATISFIES2(Y,X,Y LEXORDER X) )
   -> REPLY(N+1,L) & NREPLY(N+1) & NEGATE(ALL);

      " REPLY -> REPLYO IN V20 - V37 "

V51; "CHECK PICKUP" = CHECKPICKUP(O) -> CHECKPICKUP2(O) & NEGATE(1);
V51A; "PICKUP OK" = CHECKPICKUP2(O) & GRASPING(H,O)
      & NOT( EXISTS(R,O2,S) & HASREL(O,R,O2,S)
            & SATISFIES(R,R MEMQ '(IN ON)) )
   -> REPLYO('(OKAY)) & NEGATE(1);
V51H; "PICKUP OK" = CHECKPICKUP2(O) & NOT( EXISTS(H) & GRASPING(H,O) )
   -> REPLYO('(COULDNT GRASP)) & NEGATE(1);
V51O; "PICKUP OK" = CHECKPICKUP2(O) & GRASPING(H,O) & HASREL(O,R,O2,S)
      & SATISFIES(R,R MEMQ '(IN ON))
   -> REPLYO('(COULDNT RAISE));
V52; "CHECK PUTON" = CHECKPUTON(O,R,O2) -> CHECKPUTON2(O,R,O2) & NEGATE(1);
V52A; "PUTON OK" = CHECKPUTON2(O,R,O2) & HASREL(O,R,O2,S)
   -> REPLYO('(OKAY)) & NEGATE(1);
V52F; "PUTON OFF" = CHECKPUTON2(O,R,O2) & NOT( EXISTS(S) & HASREL(O,R,O2,S) )
   -> REPLYO('(FAILED TO PUT) @ <O,R>) & NEGATE(1);
V53; "CHECK PUTDOWN" = CHECKPUTDOWN(O,X,Y,Z)
   -> CHECKPUTDOWN2(O,X,Y,Z) & NEGATE(1);
V53D; "PUTDOWN OK" = CHECKPUTDOWN2(O,X,Y,Z) & HASREL(O,R,O2,S)
      & SATISFIES(R,R MEMQ '(IN ON)) & NOT LOCAT(O,X,Y,Z)
   -> REPLYO('(OKAY)) & NEGATE(1);
V53L; "LOC SAME" = CHECKPUTDOWN2(O,X,Y,Z) & LOCAT(O,X,Y,Z)
   -> REPLYO('(CANT MOVE IT)) & NEGATE(1);
V53N; "NOT ON" = CHECKPUTDOWN2(O,X,Y,Z) & NOT LOCAT(O,X,Y,Z)
      & NOT( EXISTS(R,O2,S) & HASREL(O,R,O2,S)
            & SATISFIES(R,R MEMQ '(IN ON)) )
   -> REPLYO('(NOT PUT DOWN)) & NEGATE(1);
V54; "CHECK STACKUP" = CHECKSTACKUP(O) -> CHECKSTACKUP2(O) & NEGATE(1);
V54A; "STACK" = CHECKSTACKUP2(O) & INSTACK(O,S)
      & NOT( EXISTS(O2) & INSTACK(O2,S) & VNEQ(O2,O)
            & SATISFIES2(O2,O,O2 LEXORDER O) )
      & NOT( EXISTS(O2) & CHECKSTACKUP2(O2) & NOT INSTACK(O2,S) )
   -> REPLYO('(OKAY)) & NEGATE(1);
V54F; "NOT ALL" = CHECKSTACKUP2(O) & INSTACK(O,S)
      & NOT( EXISTS(O2) & INSTACK(O2,S) & VNEQ(O2,O)
            & SATISFIES2(O2,O,O2 LEXORDER O) )
      & CHECKSTACKUP2(O2) & NOT INSTACK(O2,S)
   -> REPLYO('(LEFT OUT) @ <O2>) & NEGATE(1);

      " REPLY -> REPLYO IN D24, D28, D29 "

END;
```

- -

```
      % Y = EXAMPLES %                              % PAGE 8 %

EXPR MILYO(); BEGIN          PSMACRO(MILIM);

Y0; "INIT SCENE" = WBINIT(X) -> REPRHS('Y0,<<'NOT,<'WBINIT,'X>>>)
      & TRACING(TRACEPRINTM(<'Y0,"SCENE INITIALIZED">))
      & ISA('BLOCK?-1,'BLOCK) & ISA('PYRAMID?-1,'PYRAMID)
      & ISA('BLOCK?-2,'BLOCK) & ISA('PYRAMID?-2,'PYRAMID)
      & ISA('PYRAMID?-3,'PYRAMID) & ISA('BLOCK?-3,'BLOCK)
      & ISA('BLOCK?-4,'BLOCK) & ISA('BLOCK?-5,'BLOCK)
      & ISA('BOX?-1,'BOX) & ISA('TABLE?-1,'TABLE) & ISA('HAND?-1,'HAND)

      & LOCAT('BLOCK?-1,100,100,0) & LOCAT('PYRAMID?-1,100,100,100)
      & LOCAT('BLOCK?-2,400,0,0) & LOCAT('PYRAMID?-2,640,640,1)
      & LOCAT('PYRAMID?-3,500,100,200) & LOCAT('BLOCK?-3,0,300,0)
      & LOCAT('BLOCK?-4,0,240,300) & LOCAT('BLOCK?-5,300,640,0)
      & LOCAT('BOX?-1,600,600,0) & LOCAT('TABLE?-1,0,0,0)
      & LOCAT('HAND?-1,0,100,400)

      & HASREL('BLOCK?-1,'ON,'TABLE?-1,'POS)
      & HASREL('BLOCK?-2,'ON,'TABLE?-1,'POS)
      & HASREL('PYRAMID?-2,'IN,'BOX?-1,'POS)
      & HASREL('BLOCK?-5,'ON,'TABLE?-1,'POS)
      & HASREL('BLOCK?-3,'ON,'TABLE?-1,'POS)
      & HASREL('BOX?-1,'ON,'TABLE?-1,'POS)
      & HASREL('PYRAMID?-1,'ON,'BLOCK?-1,'POS)
      & HASREL('PYRAMID?-3,'ON,'BLOCK?-2,'POS)
      & HASREL('BLOCK?-4,'ON,'BLOCK?-3,'POS)

      & HASSIZE('BLOCK?-1,100,100,100) & HASSIZE('PYRAMID?-1,100,100,100)
      & HASSIZE('BLOCK?-2,200,200,200) & HASSIZE('PYRAMID?-2,300,200,200)
      & HASSIZE('PYRAMID?-3,100,100,240) & HASSIZE('BLOCK?-3,200,300,300)
      & HASSIZE('BLOCK?-4,200,200,200) & HASSIZE('BLOCK?-5,300,100,400)
      & HASSIZE('BOX?-1,800,800,1) & HASSIZE('TABLE?-1,1200,1200,0)

      & HASAV('BLOCK?-1,'COLOR,'RED,'POS)
      & HASAV('PYRAMID?-1,'COLOR,'GREEN,'POS)
      & HASAV('BLOCK?-2,'COLOR,'GREEN,'POS)
      & HASAV('PYRAMID?-2,'COLOR,'BLUE,'POS)
      & HASAV('PYRAMID?-3,'COLOR,'RED,'POS)
      & HASAV('BLOCK?-3,'COLOR,'RED,'POS)
      & HASAV('BLOCK?-4,'COLOR,'GREEN,'POS)
      & HASAV('BLOCK?-5,'COLOR,'BLUE,'POS)

      & HASAV('BLOCK?-1,'SIZE,'SMALL,'POS)
      & HASAV('PYRAMID?-1,'SIZE,'SMALL,'POS)
      & HASAV('BLOCK?-2,'SIZE,'LARGE,'POS)
      & HASAV('PYRAMID?-2,'SIZE,'LARGE,'POS)
      & HASAV('PYRAMID?-3,'SIZE,'SMALL,'POS)
      & HASAV('BLOCK?-3,'SIZE,'LARGE,'POS)
      & HASAV('BLOCK?-4,'SIZE,'LARGE,'POS)
```

```
    & HASAV('BLOCK7-5,'SIZE,'LARGE,'POS)

    & CLEARTOP('PYRAMID7-1) & CLEARTOP('PYRAMID7-2) & CLEARTOP('PYRAMID7-3)
    & CLEARTOP('BLOCK7-4) & CLEARTOP('BLOCK7-5)

    & NEGATE(1);

    % LEAVING OUT: (#IS ? COLOR) (#IS ? SHAPE) (#IS ? #ROBOT)
        (#IS ? #PERSON) (#IS ? #HAND) (#MANIP ?) (#SHAPE ?)
        (#CALL ? ?)    %

END;

END.
```

Appendix F. WBLOX PROGRAM LISTING

```
BEGIN      % PS FOR WINOGRAD'S PLANNER BLOCKS THEOREMS %

EXPR WBLOX(); BEGIN    REQUIRE(WBLOQS,WBLOW,WBLOW2,MILIPS); PSMACRO(MILIPS);

    % P GROUPS: Q, W %

    % Q Ps: BASIC BLOCKS OPERATORS %

    % MOVE HAND %

Q1: "MOVE HAND" = MOVEHAND(X,Y,Z) & ISA(H,W) & SATISFIES(W,W EQ 'HAND)
    & LOCAT(H,X1,Y1,Z1) & NOT LOCAT(H,X,Y,Z)
    & NOT( EXISTS(O) & GRASPING(H,O) ) & EVENTTIME(M)
  -> LOCAT(H,X,Y,Z) & NEGATE(1,4,7) & EVENTTIME(M+1)
    & UNEVENT(M,<MOVEHAND,X1,Y1,Z1>)
    & TRACING(TRACEPRINTM(<M>,'MOVING,'HAND,'FROM,<X1,Y1,Z1>,'TO,<X,Y,Z>));
Q2: "LIFT OBJECT" = MOVEHAND(X,Y,Z) & GRASPING(H,O) & LOCAT(O,X1,Y1,Z1)
    & HASSIZE(O,SX1,SY1,SZ1) & NOT LOCAT(H,X,Y,Z)
    & NOT( EXISTS(O2,X2,Y2,Z2,SX2,SY2,SZ2) & LOCAT(O2,X2,Y2,Z2) & VNEQ(O2,O)
        & HASSIZE(O2,SX2,SY2,SZ2)
        & SATISFIES(X, X - SX1 / 2 ?<LESS X2 + SX2)
        & SATISFIES(X, X2 ?<LESS X + SX1 / 2)
        & SATISFIES(Y, Y - SY1 / 2 ?<LESS Y2 + SY2)
        & SATISFIES(Y, Y2 ?<LESS Y + SY1 / 2)
        & SATISFIES(Z, Z - SZ1 ?<LESS Z2 + SZ2)
        & SATISFIES(Z, Z2 ?<LESS Z) )
    % A + S / 2 AND A - S / 2 GET UPPER & LOWER CORNERS, GIVEN A
        IS AT CENTER, FOR SOME DIMENSION %
    % IF FOR X, Y, Z THOSE SATISFIES'S ARE TRUE, THE OBJECT O2 OVERLAPS
        THE SPACE WHERE THE GRASPED OBJECT WOULD BE PUT;
        THE OVERLAP TESTS ARE DERIVED BY NEGATING THE NON-OVERLAP
        CONDITION, NAMELY THAT BOTH CORNERS OF ONE OBJECT
        (IN EACH DIMENSION) ARE EITHER BELOW OR ABOVE BOTH CORNERS
        OF THE OTHER; THIS IS TW'S CLEAR PREDICATE %
    & LOCAT(H,X3,Y3,Z3) & EVENTTIME(M)
  -> NEWLOCAT(O) & NEWLOCAT2(O) & LOCAT(O,X - SX1 / 2,Y - SY1 / 2,Z - SZ1)
    & TRACING(TRACEPRINTM(<M>,'LIFTING,O,'FROM,<X1,Y1,Z1>,
        'TO,<X - SX1 / 2,Y - SY1 / 2,Z - SZ1>))
    & EVENTTIME(M+1) & UNEVENT(M,<MOVEHAND,X3,Y3,Z3>) & NEGATE(1,5,7,8)
    & LOCAT(H,X,Y,Z);
Q2L: "MOVE LAP" = MOVEHAND(X,Y,Z) & GRASPING(H,O) & HASSIZE(O,SX1,SY1,SZ1)
    & LOCAT(O2,X2,Y2,Z2) & VNEQ(O2,O) & HASSIZE(O2,SX2,SY2,SZ2)
    & SATISFIES(X, X - SX1 / 2 ?<LESS X2 + SX2)
    & SATISFIES(X, X2 ?<LESS X + SX1 / 2)
    & SATISFIES(Y, Y - SY1 / 2 ?<LESS Y2 + SY2)
    & SATISFIES(Y, Y2 ?<LESS Y + SY1 / 2)
    & SATISFIES(Z, Z - SZ1 ?<LESS Z2 + SZ2) & SATISFIES(Z, Z2 ?<LESS Z)
  -> TRACING(TRACEPRINTM(<MOVE,'TO,<X,Y,Z>,'OVERLAPS,O,'WITH,O2>))
    & NEGATE(1);
Q3: "MOVE -" = MOVEHAND(X,Y,Z) & LOCAT(H,X,Y,Z) & ISA(H,W)
    & SATISFIES(W,W EQ 'HAND)
  -> NEGATE(1);

    % UPDATES FOR EFFECTS OF MOVE %

Q6: "REM ON" = NEWLOCAT(O1) & LOCAT(O1,X1,Y1,Z1) & HASREL(O1,R,O,S)
    & SATISFIES(R,R MEMQ '(IN ON))
  -> REMOHASREL(O1,R,O,S) & ERSREMOHASREL(O1,R,O,S) & NEGATE(1,3);
Q7: "ADD NEW ON" = NEWLOCAT2(O1) & LOCAT(O1,X1,Y1,Z1) & LOCAT(O2,X2,Y2,Z2)
    & ISA(O2,W) & NOT SATISFIES(W,W EQ 'PYRAMID) & VNEQ(O2,O1)
    & HASSIZE(O1,SX1,SY1,SZ1) & HASSIZE(O2,SX2,SY2,SZ2)
    & SATISFIES2(X1,X2,LESSP(X2,X1 + SX1 / 2,X2 + SX2))
    & SATISFIES2(Y1,Y2,LESSP(Y2,Y1 + SY1 / 2,Y2 + SY2))
    & SATISFIES2(Z1,Z2,Z1 EQ Z2 + SZ2)      % CHECKS O1 SUPPORTABLE %
  -> HASREL(O1,'ON,O2,'POS) & NEGATE(1) & NOT NEWLOCAT(O1);

Q11: "OFF STACK" = REMOHASREL(O,R,O2,P) & SATISFIES(R,R EQ 'ON) & INSTACK(O,S)
    & INSTACK(O2,S)
  -> REMOINSTACK(O,S) & NEGATE(3) & TRACING(TRACEPRINTM(<TAKING,O,'FROM,S>));
Q13: "KILL STACK" = REMOINSTACK(O,S) & INSTACK(O1,S)
    & NOT( EXISTS(O2) & INSTACK(O2,S) & VNEQ(O2,O1) )
  -> NEGATE(1,2) & TRACING(TRACEPRINTM(<S,'DISMANTLED>));
Q15: "ON STACK" = HASREL(O1,R,O2,P) & SATISFIES(R,R EQ 'ON) & INSTACK(O2,S)
    & NOT INSTACK(O1,S)
  -> INSTACK(O1,S) & TRACING(TRACEPRINTM(<ADDING,O1,'TO,S>));
    % EVERYTHING ON OR TRANSITIVELY-ON THE BASE BLOCK
        IS IN THE SAME STACK; MORE SOPHISTICATED PROCEDURE MIGHT
        DISTINGUISH EACH BRANCH OF "TREE" AS A SEPARATE STACK %
Q17: "NEW STACK" = HASREL(O1,R,O2,P) & SATISFIES(R,R EQ 'ON)
    & NOT( EXISTS(S) & INSTACK(O2,S) )
```

```
        & NOT( EXISTS(T1) & ISA(O2,T1) & SATISFIES(T1,T1 EQ 'TABLE) )
        & NOT( EXISTS(T1) & ISA(O2,T1) & SATISFIES(T1,T1 EQ 'BOX) )
    -> EXISTS(STACK) & INSTACK(O1,STACK) & INSTACK(O2,STACK)
        & TRACING(TRACEPRINTM(<'MAKING,'STACK,STACK,O1,O2>));

Q21; "ON BOX" = HASREL(O1,R,O2,S) & SATISFIES(R,R EQ 'ON) & ISA(O2,W)
        & SATISFIES(W,W EQ 'BOX)
    -> HASREL(O1,'IN,O2,S) & NEGATE(1) & NOT CLEARTOP(O2);
Q23; "OFF CLEAR" = REMOHASREL(O1,R,O2,S) & SATISFIES(R,R MEMQ '(IN ON))
        & NOT( EXISTS(O3) & HASREL(O3,R,O2,S) )
    -> CLEARTOP(O2);
Q27; "ON -CLEAR" = HASREL(O1,R,O2,S) & SATISFIES(R,R EQ 'ON) & CLEARTOP(O2)
        & ISA(O2,W) & NOT SATISFIES(W,W EQ 'BOX)
    -> NEGATE(3);
Q29; "ERS REM" = ERSREMOHASREL(O1,R,O2,S)
    -> NOT REMOHASREL(O1,R,O2,S) & NEGATE(1);


                          ......................


        % PUT %                             % PAGE 2 %

Q31; "PUT" = PUT(G1,O,X,Y,Z) & HASLEVEL(G1,N)
    -> EXISTS(G) & GRASP(G,O) & NEXT(G,<'PUTMOVE,G1,O,X,Y,Z>) & HASLEVEL(G,N+1)
        & NEGATE(1) & TRACING(TRACEPRINTG(<G,'GRASP,O>,N+1));
Q32; "PUT MOVE" = PUTMOVE(G,O,X,Y,Z) & HASSIZE(O,SX,SY,SZ)
    -> MOVEHAND(X + SX / 2,Y + SY / 2, Z + SZ) & UNGRASP(O) & SUCCEED(G)
        & NEGATE(1);
        % ASSUMES CLEAR AND SUPPORT ARE CHECKED BY MOVEHAND %

        % RAISE HAND %

Q35; "RAISE HAND" = RAISEHAND(H) & LOCAT(H,X,Y,Z)
    -> MOVEHAND(X,Y,1700) & NEGATE(1);

        % GRASP %

Q41; "GRASPING" = GRASP(G,O) & GRASPING(H,O) -> SUCCEED(G) & NEGATE(1);
Q43; "GRASP HOLDING" = GRASP(G1,O) & GRASPING(H,O2) & VNEQ(O,O2)
        & HASLEVEL(G1,N)
    -> EXISTS(G) & GETRIDOF(G,O2) & NEXT(G,<'GRASP,G1,O>) & HASLEVEL(G,N+1)
        & NEGATE(1) & TRACING(TRACEPRINTG(<G,'GETRIDOF,O2>,N+1));
Q45; "GRASP" = GRASP(G1,O) & NOT( EXISTS(H,O2) & GRASPING(H,O2) )
        & LOCAT(O,X,Y,Z) & HASSIZE(O,SX,SY,SZ) & HASLEVEL(G1,N)
    -> EXISTS(G) & CLEAROFF(G,O)
        & NEXT(G,<GRASP1,G1,O,X + SX / 2,Y + SY / 2,Z + SZ>) & HASLEVEL(G,N+1)
        & NEGATE(1) & TRACING(TRACEPRINTG(<G,'CLEAROFF,O>,N+1));
Q46; "GRASP MOVE" = GRASP1(G,O,X,Y,Z)
    -> MOVEHAND(X,Y,Z) & GRASP2(G,O) & NEGATE(1);
Q47; "GRASP ACT" = GRASP2(G,O) & ISA(H,W) & SATISFIES(W,W EQ 'HAND)
        & EVENTTIME(M)
    -> SUCCEED(G) & GRASPING(H,O) & NEGATE(1,A) & UNEVENT(M,<UNGRASP,O>)
        & EVENTTIME(M+1) & TRACING(TRACEPRINTM(<M>,'GRASPING,O>));
Q47U; "BACKING GRASP" = GRASP3(H,O) & EVENTTIME(M)
    -> GRASPING(H,O) & EVENTTIME(M+1) & UNEVENT(M,<UNGRASP,O>)
        & TRACING(TRACEPRINTM(<M>,'GRASPING,O>)) & NEGATE(ALL);

        % UNGRASP %

Q49; "UNGRASP" = UNGRASP(O) & GRASPING(H,O) & HASREL(O,R,O2,S)
        & SATISFIES(R,R MEMQ '(ON IN)) & EVENTTIME(M)
    -> NEGATE(1,2,5) & TRACING(TRACEPRINTM(<M>,'LETTING,'GO,'OF,O>))
        & UNEVENT(M,<GRASP3,H,O>) & EVENTTIME(M+1);

    END;


                          ......................


EXPR WBLOQ5(); BEGIN                        % PAGE 3 %

        % FIND SPACE %

Q51; "FIND & CENTER" = FINDSPACE(O,I,SX,SY,SZ) & ISA(O,W)
        % FIND SPACE ON O, IGNORING I, SIZE TRIPLE <SX,SY,SZ> %
        & NOT SATISFIES(W,W EQ 'BOX) & NOT SATISFIES(W,W EQ 'TABLE)
        & NOT( EXISTS(A) & NOCLEAR(A) )
    -> LOCATESPACE(O,I,SX,SY,SZ) & USERESULT(O,I,SX,SY,'CENTER) & NEGATE(1);
Q52; "FIND & PACK BOX" = FINDSPACE(O,I,SX,SY,SZ) & ISA(O,W)
        & SATISFIES(W,W EQ 'BOX)
    -> LOCATESPACE(O,I,SX,SY,SZ) & USERESULT(O,I,SX,SY,'PACK) & NEGATE(1);
Q53; "FIND & PACK NOCLEAR" = FINDSPACE(O,I,SX,SY,SZ) & ISA(O,W)
        & SATISFIES(W,W NEQ 'TABLE) & NOCLEAR(A)
    -> LOCATESPACE(O,I,SX,SY,SZ) & USERESULT(O,I,SX,SY,'PACK) & NEGATE(1);
Q54; "FIND RANDOM" = FINDSPACE(O,I,SX,SY,SZ) & ISA(O,W)
        & SATISFIES(W,W EQ 'TABLE)
        % THAT IS THE RESULT OF CONJOINING THE NEGATED CONDITIONS OF Q51-Q53;
            -51 = BOX OR TABLE OR NOCLEAR; -52 = -BOX;
             -53 = TABLE OR -NOCLEAR; 2 SIMPLE RESOLUTIONS SIMPLIFIES IT %
    -> LOCATESPACE(O,I,SX,SY,SZ) & USERESULT(O,I,SX,SY,'RANDOM) & NEGATE(1);

Q57; "LOCATE CLEAR" = LOCATESPACE(O,I,SX,SY,SZ) & CLEARTOP(O)
        & LOCAT(O,X1,Y1,Z1) & HASSIZE(O,SX1,SY1,SZ1)
        & SATISFIES3(SX,SY,SX1,NOT(SX %GREAT SX1) & NOT(SY %GREAT SY1))
    -> LOCATERESULT(I,X1,Y1,X1 + SX1,Y1 + SY1,Z1 + SZ1) & NEGATE(1)
        & TRACING(TRACEPRINTM(<'FOUND,'REGION,'CLEARTOP,O>));
Q57F; "LOCATE NO FIT" = LOCATESPACE(O,I,SX,SY,SZ) & CLEARTOP(O)
        & HASSIZE(O,SX1,SY1,SZ1)
        & NOT SATISFIES3(SX,SY,SX1,NOT(SX %GREAT SX1) & NOT(SY %GREAT SY1))
        & USERESULT(O,I,SX,SY,U)
    -> FAILLOCATE(I) & NEGATE(1,5)
        & TRACING(TRACEPRINTM(<'FINDSPACE,'CLEARTOP,O,'TOO,'SMALL>));
        % NEED ? SHORTCUT: ONLY THE IGNORED OBJECT ITSELF IS ON TOP %

Q61; "LOCATE START" = LOCATESPACE(O,I,SX,SY,SZ) & NOT CLEARTOP(O)
        & LOCAT(O,X1,Y1,Z1) & HASSIZE(O,SX1,SY1,SZ1)
    -> FINDLOWPAIR(10,I,X1,Y1,X1 + SX1,Y1 + SY1,Z1 + SZ1,
            RANDOM(X1,X1 + SX1 - (2 * SX / 3)),
            RANDOM(Y1,Y1 + SY1 - (2 * SY / 3)),SX,SY,SZ)
        & NEGATE(1);

        % THE FOLLOWING ATTEMPTS TO FIND A REGION AROUND A RANDOM POINT
            BIG ENOUGH TO ENCLOSE THE REQUIRED SPACE;
            IT WILL NOT IN EVERY CASE FIND THE MAXIMAL REGION GIVEN A
            POINT, BUT FOR EVERY MAX REGION, THERE IS A POINT SUCH THAT
            THE ALGORITHM WOULD GET TO THAT REGION FROM THE POINT;
            IT DOES NOT FIND THE BOUNDARY OF THE CLEAR REGION AROUND
            THE POINT, BUT RATHER "GROPES" AROUND THE POINT FORMING
            PROVISIONAL BOUNDARY CORNERS USING COORDINATES OF THE
            CLOSEST OBJECTS, CONSIDERED INDEPENDENTLY %

Q62; "LOW PAIR" = FINDLOWPAIR(N,O,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & SATISFIES(N,N %GREAT 0)
        & NOT( EXISTS(O2,X3,Y3,SX3,SY3,SZ3) & LOCAT(O2,X3,Y3,Z) & VNEQ(O2,O)
            & HASSIZE(O2,SX3,SY3,SZ3)
            & SATISFIES3(X0,X3,SX3,LESSP(X3,X0,X3 + SX3))
            & SATISFIES3(Y0,Y3,SY3,LESSP(Y3,Y0,Y3 + SY3))
            & SATISFIES(SZ3,SZ3 %GREAT 0) )
        % <X0,Y0> NOT INSIDE SOME OBJECT %
    -> FINDLOWX(O,X1,X0,Y1,Y2,Z) & FINDLOWY(O,X1,X2,Y1,Y0,Z) & LOWX(N,O,X1)
        & LOWY(N,O,Y1) & GROWTOFIT(N,O,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & NEGATE(1) & TRACING(TRACEPRINTM(<'LOOKING,'AT,<X0,Y0,Z>>));
Q63; "LOCATE EXH" = FINDLOWPAIR(N,O,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & SATISFIES(N,N EQ 0) & USERESULT(O2,O,SX,SY,U)
    -> FAILLOCATE(O) & NEGATE(1,3)
        & TRACING(TRACEPRINTM(<'FINDSPACE,'LIMIT,'EXCEEDED>));
Q64; "RANDOM OBSTR" = FINDLOWPAIR(N,O,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & SATISFIES(N,N %GREAT 0) & LOCAT(O2,X3,Y3,Z) & VNEQ(O2,O)
        & HASSIZE(O2,SX3,SY3,SZ3)
        & SATISFIES3(X0,X3,SX3,LESSP(X3,X0,X3 + SX3))
        & SATISFIES3(Y0,Y3,SY3,LESSP(Y3,Y0,Y3 + SY3))
        & SATISFIES(SZ3,SZ3 %GREAT 0)
    -> FINDNEARPAIR(N,O,X3,Y0) & FINDNEARPAIR(N,O,X3 + SX3,Y0)
        & FINDNEARPAIR(N,O,X0,Y3) & FINDNEARPAIR(N,O,X0,Y3 + SY3)
        & TRACING(TRACEPRINTM(<'REJECTING,<X0,Y0,Z>>));
Q64A; "NEAR SEL" = FINDNEARPAIR(N,O,X,Y)
        & FINDLOWPAIR(N,O,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & SATISFIES3(X1,X2,X,LESSP(X1,X,X2))
        & SATISFIES3(Y1,Y2,Y,LESSP(Y1,Y,Y2))
        & NOT( EXISTS(X3,Y3) & FINDNEARPAIR(N,O,X3,Y3)
            & SATISFIES3(X1,X2,X3,LESSP(X1,X3,X2))
            & SATISFIES3(Y1,Y2,Y3,LESSP(Y1,Y3,Y2))
            & SATISFIES3(X3,Y3,X0,MAX(ABS(X0-X3),ABS(Y0-Y3))
                %LESS MAX(ABS(X0-X),ABS(Y0-Y))) )
        & NOT( EXISTS(X3,Y3) & FINDNEARPAIR(N,O,X3,Y3)
            & SATISFIES3(X1,X2,X3,LESSP(X1,X3,X2))
            & SATISFIES3(Y1,Y2,Y3,LESSP(Y1,Y3,Y2))
            & SATISFIES3(X3,Y3,X0,MAX(ABS(X0-X3),ABS(Y0-Y3))
                = MAX(ABS(X0-X),ABS(Y0-Y)))
            & SATISFIES3(X,Y,X3,1000 * X3 + Y3 %LESS 1000 * X + Y) )
    -> ERSFINDNEARPAIR(N,O) & FINDLOWPAIR(N-1,O,X1,Y1,X2,Y2,Z,X,Y,SX,SY,SZ)
        & NEGATE(1,2);
Q64B; "FILLED" = FINDNEARPAIR(N,O,X,Y)
        & FINDLOWPAIR(N,O,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & NOT( EXISTS(X3,Y3) & FINDNEARPAIR(N,D,X3,Y3)
              & SATISFIES3(X1,X2,X3,LESSP(X1,X3,X2))
              & SATISFIES3(Y1,Y2,Y3,LESSP(Y1,Y3,Y2)) )
        & NOT( EXISTS(X3,Y3) & FINDNEARPAIR(N,D,X3,Y3)
              & SATISFIES2(X,X3,X3 ?=LESS X) )
        & USERESULT(O7,D,SX,SY,U)
     -> ERSFINDNEARPAIR(N,D) & FAILLOCATE(0) & NEGATE(ALL)
        & TRACING(TRACEPRINTM(<'SPACE,'FILLED,<X1,Y1,Z>,<X2,Y2,Z>>));
Q64E; "ERS NEAR" = ERSFINDNEARPAIR(N,D) & FINDNEARPAIR(N,D,X,Y) -> NEGATE(ALL);

Q65; "LOW X" = FINDLOWX(O,X1,X2,Y1,Y2,Z) & LOWX(N,D,X1) & LOCAT(O3,X3,Y3,Z)
        & VNEQ(O3,D) & HASSIZE(O3,SX3,SY3,SZ3)
        & SATISFIES3(X1,X2,X3,NOT(X1 ?=GREAT X3 + SX3)
              & NOT(X3 + SX3 ?=GREAT X2))
        & SATISFIES3(Y1,Y2,Y3,NOT(Y1 ?=GREAT Y3 + SY3)
              & NOT(Y3 + SY3 ?=GREAT Y2))
        & NOT( EXISTS(O4,X4,Y4,SX4,SY4,SZ4) & LOCAT(O4,X4,Y4,Z) & VNEQ(O4,D)
              & VNEQ(O4,O3) & HASSIZE(O4,SX4,SY4,SZ4)
              & SATISFIES3(X3,X2,X4,X3 + SX3 ?=LESS X4 + SX4
                    & NOT(X4 + SX4 ?=GREAT X2))
              & SATISFIES3(Y3,Y2,Y4,NOT(Y1 ?=GREAT Y4 + SY4)
                    & NOT(Y4 + SY4 ?=GREAT Y2)) )
        % DON'T HAVE TO DO MORE GENERAL OVERLAP ON Y DIMENSION BECAUSE
              ASSUMING FULL OBJECT HAS TO BE ON WHATEVER WE'RE FINDING
              SPACE ON %
        % DON'T CARE IF MULTIPLE FIRES (TIES) - RHS SAME ANYWAY %
     -> LOWX(N,D,X3 + SX3) & NEGATE(2);
Q66; "LOW Y" = FINDLOWY(O,X1,X2,Y1,Y2,Z) & LOWY(N,D,Y1) & LOCAT(O3,X3,Y3,Z)
        & VNEQ(O3,D) & HASSIZE(O3,SX3,SY3,SZ3)
        & SATISFIES3(X1,X2,X3,NOT(X1 ?=GREAT X3 + SX3)
              & NOT(X3 + SX3 ?=GREAT X2))
        & SATISFIES3(Y1,Y2,Y3,NOT(Y1 ?=GREAT Y3 + SY3)
              & NOT(Y3 + SY3 ?=GREAT Y2))
        & NOT( EXISTS(O4,X4,Y4,SX4,SY4,SZ4) & LOCAT(O4,X4,Y4,Z) & VNEQ(O4,D)
              & VNEQ(O4,O3) & HASSIZE(O4,SX4,SY4,SZ4)
              & SATISFIES3(X3,X2,X4,NOT(X1 ?=GREAT X4 + SX4)
                    & NOT(X4 + SX4 ?=GREAT X2))
              & SATISFIES3(Y3,Y2,Y4,Y3 + SY3 ?=LESS Y4 + SY4
                    & NOT(Y4 + SY4 ?=GREAT Y2)) )
        % DON'T HAVE TO DO MORE GENERAL OVERLAP ON X DIMENSION BECAUSE
              ASSUMING FULL OBJECT HAS TO BE ON WHATEVER WE'RE FINDING
              SPACE ON %
        % DON'T CARE IF MULTIPLE FIRES (TIES) - RHS SAME ANYWAY %
     -> LOWY(N,D,Y3 + SY3) & NEGATE(2);

Q67; "GROW READY" = GROWTOFIT0(N,D,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
     -> GROWTOFIT(N,D,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & CHECKFAILFIT(N,D,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ) & NEGATE(1);
Q68; "SIZES FIT" = GROWTOFIT(N,D,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & LOWX(N,D,X) & LOWY(N,D,Y)
        & NOT( EXISTS(O3,X3,Y3,SX3,SY3,SZ3) & LOCAT(O3,X3,Y3,Z) & VNEQ(O3,D)
              & HASSIZE(O3,SX3,SY3,SZ3) & SATISFIES(SZ3,SZ3 ?=GREAT 0)
              & SATISFIES(X, X ?=LESS X3 + SX3)
              & SATISFIES(X, X3 ?=LESS X + SX)
              & SATISFIES(Y, Y ?=LESS Y3 + SY3)
              & SATISFIES(Y, Y3 ?=LESS Y + SY) )
        % NO OVERLAP WITHIN THE DESIRED SIZE; CF. Q7 %
        & NOT SATISFIES3(X,SX,X2,X + SX ?=GREAT X2)
        & NOT SATISFIES3(Y,SY,Y2,Y + SY ?=GREAT Y2)
        & FINDLOWX(O,A,B,C,D,Z) & FINDLOWY(O,E,F,G,H,Z)
     -> FINDHIGHX(O,X + SX,X2,Y,Y2,Z) & FINDHIGHY(O,X,X2,Y + SY,Y2,Z)
        % TRYING TO PUSH OUT REGION FURTHER %
        & FOUNDHIGHPAIR0(N,D,X,Y,Z) & HIGHX(N,D,X2) & HIGHY(N,D,Y2)
        & NEGATE(ALL) & NOT CHECKFAILFIT(N,D,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ);
Q69; "FIT FAIL" = CHECKFAILFIT(N,D,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & FINDLOWX(O,A,B,C,D,Z) & FINDLOWY(O,E,F,G,H,Z) & LOWX(N,D,I)
        & LOWY(N,D,J)
     -> FINDLOWPAIR(N - 1,D,X1,Y1,X2,Y2,Z,RANDOM(X1,X2 - (2 * SX / 3)),
              RANDOM(Y1,Y2 - (2 * SY / 3)),SX,SY,SZ)
        & NEGATE(ALL) & NOT GROWTOFIT(N,D,X1,Y1,X2,Y2,Z,X0,Y0,SX,SY,SZ)
        & TRACING(TRACEPRINTM(<'REGION,'AT,<I,J,Z>,'TOO,'SMALL>));

Q70; "HIGH X" = FINDHIGHX(O,X1,X2,Y1,Y2,Z) & HIGHX(N,D,X2) & LOCAT(O3,X3,Y3,Z)
        & VNEQ(O3,D) & HASSIZE(O3,SX3,SY3,SZ3)
        % WANT MIN X OF OBJECT THAT OVERLAPS THE Y DIMENSION %
        & SATISFIES2(X1,X3,NOT(X1 ?=GREAT X3))
        & SATISFIES3(Y1,Y3,SY3,Y1 ?=LESS Y3 + SY3)
        & SATISFIES2(Y2,Y3,Y3 ?=LESS Y2)        % CF. Q2 %
        & NOT( EXISTS(O4,X4,Y4,SX4,SY4,SZ4) & LOCAT(O4,X4,Y4,Z)
              & VNEQ(O4,O3) & VNEQ(O4,D) & HASSIZE(O4,SX4,SY4,SZ4)
              & SATISFIES3(X1,X3,X4,NOT(X1 ?=GREAT X4) & X4 ?=LESS X3)
              & SATISFIES3(Y1,Y4,SY4,Y1 ?=LESS Y4 + SY4)
              & SATISFIES2(Y2,Y4,Y4 ?=LESS Y2) )
        % NEED GENERAL OVERLAP BECAUSE COMPARING IN RESTRICTED REGION; CF. Q65 %
        % DON'T CARE IF MULTIPLE FIRINGS, BECAUSE RHS SAME %
     -> HIGHX(N,D,X3) & NEGATE(2);
Q71; "HIGH Y" = FINDHIGHY(O,X1,X2,Y1,Y2,Z) & HIGHY(N,D,Y2) & LOCAT(O3,X3,Y3,Z)
        & VNEQ(O3,D) & HASSIZE(O3,SX3,SY3,SZ3)
        % WANT MIN Y OF OBJECT THAT OVERLAPS THE X DIMENSION %
        & SATISFIES2(Y1,Y3,NOT(Y1 ?=GREAT Y3))
        & SATISFIES3(X1,X3,SX3,X1 ?=LESS X3 + SX3)
        & SATISFIES2(X2,X3,X3 ?=LESS X2)        % CF. Q2 %
        & NOT( EXISTS(O4,X4,Y4,SX4,SY4,SZ4) & LOCAT(O4,X4,Y4,Z)
              & VNEQ(O4,O3) & VNEQ(O4,D) & HASSIZE(O4,SX4,SY4,SZ4)
              & SATISFIES3(Y1,Y3,Y4,NOT(Y1 ?=GREAT Y4) & Y4 ?=LESS Y3)
              & SATISFIES3(X1,X4,SX4,X1 ?=LESS X4 + SX4)
              & SATISFIES2(X2,X4,X4 ?=LESS X2) )
        % NEED GENERAL OVERLAP BECAUSE COMPARING IN RESTRICTED REGION; CF. Q65 %
        % DON'T CARE IF MULTIPLE FIRINGS, BECAUSE RHS SAME %
     -> HIGHY(N,D,Y3) & NEGATE(2);

Q72; "HIGH READY" = FOUNDHIGHPAIR0(N,D,X,Y,Z)
     -> FOUNDHIGHPAIR(N,D,X,Y,Z) & NEGATE(1);
Q73; "HIGH PAIR" = FOUNDHIGHPAIR(N,D,X,Y,Z) & HIGHX(N,D,X1) & HIGHY(N,D,Y1)
        & FINDHIGHX(O,A,B,C,D,Z) & FINDHIGHY(O,E,F,G,H,Z)
     -> LOCATERESULT(O,X,Y,X1,Y1,Z) & NEGATE(ALL)
        & TRACING(TRACEPRINTM(<'FOUND,'REGION,<X,Y,Z>,'TO,<X1,Y1,Z>>));

Q76; "LOCATE CENTER" = LOCATERESULT(O,X1,Y1,X2,Y2,Z) & USERESULT(O1,D,SX,SY,U)
        & SATISFIES(U,U EQ 'CENTER)
     -> FOUNDSPACE(O1,D,(X1 + X2 - SX) / 2,(Y1 + Y2 - SY) / 2,Z) & NEGATE(ALL);
Q77; "LOCATE PACK" = LOCATERESULT(O,X1,Y1,X2,Y2,Z) & USERESULT(O1,D,SX,SY,U)
        & SATISFIES(U,U EQ 'PACK)
     -> FOUNDSPACE(O1,D,X1,Y1,Z) & NEGATE(ALL);
Q78; "LOCATE RANDOM" = LOCATERESULT(O,X1,Y1,X2,Y2,Z) & USERESULT(O1,D,SX,SY,U)
        & SATISFIES(U,U EQ 'RANDOM)
     -> FOUNDSPACE(O1,D,RANDOM(X1,X2 - SX),RANDOM(Y1,Y2 - SY),Z) & NEGATE(ALL);

        % MAKE SPACE %

Q81; "MAKE SPACE" = MAKESPACE(G1,D,1,SX,SY,SZ) & HASREL(O2,R,D,B)
        & SATISFIES(R,R EQ 'ON) & HASSIZE(O2,SX2,SY2,SZ2)
        & SATISFIES3(SX,SY,SX2,NOT(SX2 ?=LESS SX) & NOT(SY2 ?=LESS SY))
        & NOT( EXISTS(O3,SX3,SY3,SZ3) & HASREL(O3,R,D,B)
              & HASSIZE(O3,SX3,SY3,SZ3)
              & SATISFIES3(SX,SY,SX3,NOT(SX3 ?=LESS SX)
                    & NOT(SY3 ?=LESS SY))
              & SATISFIES3(SX2,SY2,SX3,SX3 + SY3 ?=LESS SX2 + SY2) )
              % SMALLEST ONE THAT'S BIG ENOUGH %
        & NOT( EXISTS(O3,SX3,SY3,SZ3) & HASREL(O3,R,D,B)
              & HASSIZE(O3,SX3,SY3,SZ3)
              & SATISFIES3(SX,SY,SX3,NOT(SX3 ?=LESS SX)
                    & NOT(SY3 ?=LESS SY))
              & SATISFIES3(SX2,SY2,SX3,SX3 + SY3 = SX2 + SY2)
              & VNEQ(O3,O2) & SATISFIES2(O3,O2,O3 LEXORDER O2) )
              % AMONG TIES, USE LEXORDER %
        & HASLEVEL(G1,N)
     -> EXISTS(G) & GETRIDOF(G,O2) & NEXT(G,<'MAKESPACE2,G1,D,1,SX,SY,SZ>)
        & HASLEVEL(G,N+1) & TRACING(TRACEPRINTG(<G,'GETRIDOF,O2>,N+1))
        & NEGATE(1);
Q82; "MAKE SPACE M" = MAKESPACE(G1,D,1,SX,SY,SZ) & HASREL(O2,R,D,B)
        & SATISFIES(R,R EQ 'ON) & HASSIZE(O2,SX2,SY2,SZ2)
        & NOT( EXISTS(O3,SX3,SY3,SZ3) & HASREL(O3,R,D,S)
              & HASSIZE(O3,SX3,SY3,SZ3)
              & SATISFIES3(SX,SY,SX3,NOT(SX3 ?=LESS SX)
                    & NOT(SY3 ?=LESS SY)) )
        & NOT( EXISTS(O3,SX3,SY3,SZ3) & HASREL(O3,R,D,S)
              & HASSIZE(O3,SX3,SY3,SZ3)
              & SATISFIES3(SX2,SY2,SX3,SX3 + SY3 ?=GREAT SX2 + SY2) )
        & NOT( EXISTS(O3,SX3,SY3,SZ3) & HASREL(O3,R,D,S)
              & HASSIZE(O3,SX3,SY3,SZ3)
              & SATISFIES3(SX2,SY2,SX3,SX3 + SY3 = SX2 + SY2)
              & VNEQ(O3,O2) & SATISFIES2(O3,O2,O3 LEXORDER O2) )
        & HASLEVEL(G1,N)
     -> EXISTS(G) & GETRIDOF(G,O2) & NEXT(G,<'MAKESPACE2,G1,D,1,SX,SY,SZ>)
        & HASLEVEL(G,N+1) & TRACING(TRACEPRINTG(<G,'GETRIDOF,O2>,N+1))
        & NEGATE(1);
Q83; "MAKE SPACE FIND" = MAKESPACE2(G,D,1,SX,SY,SZ)
     -> FINDSPACE(D,1,SX,SY,SZ) & MAKESPACE3(G,D,1,SX,SY,SZ) & NEGATE(1);
Q84; "MAKE SPACE FND+" = MAKESPACE3(G,D,1,SX,SY,SZ) & FOUNDSPACE(O,1,X,Y,Z)
     -> SUCCEED(G) & NEGATE(1);
Q85; "MAKE SPACE FND-" = MAKESPACE3(G,D,1,SX,SY,SZ) & FAILLOCATE(1)
     -> MAKESPACE(G,D,1,SX,SY,SZ) & NEGATE(ALL);
        % (WON'T TRY TO MAKE SPACE AT ALL IF NOT ENOUGH SPACE THERE) %

END;
```

```
                    ....................

EXPR WBLOW(); BEGIN                              % PAGE 4 %

    % W P: BLOCKS OPERATORS THAT USE THE BASIC ONES - THEY PROCESS
          COMMANDS FROM THE NL FRONT END %

    % GOAL EXECUTIVE %

W0: "SUCC NEXT" = SUCCEED(G) & NEXT(G,C) & HASLEVEL(G,N)
  -> DELAYEXPND(MAKEINSTL(C)) & NEGATE(1)
    & TRACING(TRACEPRINTH(<G,'SUCCEEDS>,N+1));
  % MAKEINSTL CONVERTS THE LIST VALUE BOUND TO C TO BE AN INSTANCE
    APPROPRIATE TO BE ASSERTED; THE DELAYEXPND OPERATION IS AS IF
    THAT INSTANCE WERE TEMPORARILY SUBSTITUTED IN AS PART OF THE
    RHS; CF. W1 FOR A USE OF NEXT IN AN RHS %
W0F: "FAIL NEXT" = FAIL(G) & NEXTF(G,C) & HASLEVEL(G,N)
  -> DELAYEXPND(MAKEINSTL(C)) & NEGATE(1)
    & TRACING(TRACEPRINTH(<G,'FAILS>,N+1));
W0G: "FAIL TOP" = FAIL(G) & NOT( EXISTS(C) & NEXTF(G,C) ) & HASLEVEL(G,N)
  -> TRACING(TRACEPRINTH(<G,'FAILS,'NO,'NEXT>,N+1)) & NEGATE(1);
W0S: "SUCC SUPER" = SUCCEED(G) & NOT( EXISTS(C) & NEXT(G,C) )
    & HASSUPERGOAL(G,G2) & HASLEVEL(G,N)
  -> SUCCEED(G2) & TRACING(TRACEPRINTH(<G,'SUCCEEDS>,N+1)) & NEGATE(1);
W0T: "SUCC TOP" = SUCCEED(G) & NOT( EXISTS(C) & NEXT(G,C) )
    & NOT( EXISTS(G2) & HASSUPERGOAL(G,G2) ) & HASLEVEL(G,N)
  -> TRACING(TRACEPRINTH(<G,'SUCCEEDS>,N+1)) & NEGATE(ALL);

    % PICK UP %

W1: "PICK UP" = PICKUP(GT,O) & ISA(H,W) & SATISFIES(W,W EQ 'HAND)
    & HASLEVEL(GT,N)
  -> EXISTS(G) & GRASP(G,O) & HASLEVEL(G,N+1) & NEXT(G,<'PICKUP2,GT,H>)
    & TRACING(TRACEPRINTM(<'STARTING,GT,'PICKUP,O>))
    & NEGATE(1) & TRACING(TRACEPRINTG(<G,'GRASP,O>,N+1));
W2: "PICKUP RAISE" = PICKUP2(G,H) -> RAISEHAND(H) & SUCCEED(G) & NEGATE(1);

    % CLEAR OFF TOP OF OBJECT %

W3: "CLEAR OFF" = CLEAROFF(G1,O) & HASREL(O1,R,O,S) & SATISFIES(R,R EQ 'ON)
    & HASSIZE(O1,SX,SY,SZ) & NOT( EXISTS(O2,SX2,SY2,SZ2) & HASREL(O2,R,O,S)
        & VNEQ(O2,O1) & HASSIZE(O2,SX2,SY2,SZ2)
        & SATISFIES2(SX,SX2,SX + SY ?>GREAT SX2 + SY2) )
    & NOT( EXISTS(O2,SX2,SY2,SZ2) & HASREL(O2,R,O,S)
        & VNEQ(O2,O1) & HASSIZE(O2,SX2,SY2,SZ2)
        & SATISFIES2(SX,SX2,SX + SY = SX2 + SY2)
        & SATISFIES2(O1,O2,O1 LEXORDER O2) )
    & HASLEVEL(G1,N)
  -> EXISTS(G) & GETRIDOF(G,O1) & HASLEVEL(G,N+1) & NEXT(G,<CLEAROFF,G1,O>)
    & NEGATE(1) & TRACING(TRACEPRINTG(<G,'GETRIDOF,O1>,N+1));
    % ITERATES UNTIL ALL CLEAR %
W4: "CLEAR OFF" = CLEAROFF(G1,O) & HASREL(O1,R,O,S) & SATISFIES(R,R EQ 'IN)
    & HASSIZE(O1,SX,SY,SZ) & NOT( EXISTS(O2,SX2,SY2,SZ2) & HASREL(O2,R,O,S)
        & VNEQ(O2,O1) & HASSIZE(O2,SX2,SY2,SZ2)
        & SATISFIES2(SX,SX2,SX + SY ?>GREAT SX2 + SY2) )
    & NOT( EXISTS(O2,SX2,SY2,SZ2) & HASREL(O2,R,O,S)
        & VNEQ(O2,O1) & HASSIZE(O2,SX2,SY2,SZ2)
        & SATISFIES2(SX,SX2,SX + SY = SX2 + SY2)
        & SATISFIES2(O1,O2,O1 LEXORDER O2) )
    & HASLEVEL(G1,N)
  -> EXISTS(G) & GETRIDOF(G,O1) & HASLEVEL(G,N+1) & NEXT(G,<CLEAROFF,G1,O>)
    & NEGATE(1) & TRACING(TRACEPRINTG(<G,'GETRIDOF,O1>,N+1));
    % ITERATES UNTIL ALL CLEAR %
W5: "CLEAR -" = CLEAROFF(G,O) & CLEARTOP(O) -> SUCCEED(G) & NEGATE(1);

    % PUT DOWN %

W10: "PUT DOWN" = PUTDOWN(GT,O) & HASLEVEL(GT,N)
  -> EXISTS(G) & GETRIDOF(G,O) & HASLEVEL(G,N+1) & HASSUPERGOAL(G,GT)
    & NEGATE(1) & TRACING(TRACEPRINTM(<'STARTING,GT,'PUT,O,'DOWN>))
    & TRACING(TRACEPRINTG(<G,'GETRIDOF,O>,N+1));

    % GET RID OF %

    % THIS IS A MIXTURE OF ACTUAL PLANNER & WHAT'S GIVEN IN TW'S BOOK %

W11: "GET RID OF START" = GETRIDOF(G,O) & NOT RETRY(G) & ISA(O2,W)
    & SATISFIES(W,W EQ 'TABLE) & HASSIZE(O,SX,SY,SZ)
  -> FINDSPACE(O2,O,SX,SY,SZ) & GETRIDPUT(G,O,O2) & NEGATE(1);
W12: "GET RID FND" = GETRIDPUT(G1,O,O2) & NOT RETRY(G1) & FOUNDSPACE(O2,O,X,Y,Z)
    & HASLEVEL(G1,N) & CHOICECOUNT(K) & EVENTTIME(M)
  -> EXISTS(G) & PUT(G,O,X,Y,Z) & HASLEVEL(G,N+1) & HASSUPERGOAL(G,G1)
    & NEGATE(1,3,5) & CHOICECOUNT(K+1) & GETRIDCHOICE(K+1,G,1,O2,O,X,Y,Z)
    & CHOICETIME(K+1,M) & TRACING(TRACEPRINTG(<G,'PUT,O,OK,Y,Z>,N+1));
W13: "GET RID RETRY T" = GETRIDOF(G,O) & RETRY(G) & CHOICECOUNT(N)
    & NOT( EXISTS(L,X,Y,Z,O3) & GETRIDCHOICE(N,G,L,O3,O,X,Y,Z)
        & SATISFIES(L,L EQ 3) ) % TRY ONLY 3 TIMES ON TABLE %
    & ISA(O2,W) & SATISFIES(W,W EQ 'TABLE) & HASSIZE(O,SX,SY,SZ)
  -> FINDSPACE(O2,O,SX,SY,SZ) & GETRIDPUT(G,O,O2) & NEGATE(1);
W14: "GET RID FND RE" = GETRIDPUT(G1,O,O2) & RETRY(G1) & FOUNDSPACE(O2,O,X,Y,Z)
    & HASLEVEL(G1,N) & CHOICECOUNT(K) & GETRIDCHOICE(K,G1,J,O3,O,X2,Y2,Z2)
    & NOT( EXISTS(O4,X3,Y3,Z3,I) & GETRIDCHOICE(K,G1,I,O4,O,X3,Y3,Z3)
        & SATISFIES2(I,J,I ?>GREAT J) )
    & NOT( EXISTS(I) & GETRIDCHOICE(K,G1,I,O2,O,X,Y,Z) )
  -> EXISTS(G) & PUT(G,O,X,Y,Z) & HASLEVEL(G,N+1) & HASSUPERGOAL(G,G1)
    & NEGATE(1,3) & GETRIDCHOICE(K,G1,J+1,O2,O,X,Y,Z)
    & TRACING(TRACEPRINTG(<G,'PUT,O,OK,Y,Z>,N+1));
W14D: "GET RID FND DUPL" = GETRIDPUT(G1,O,O2) & RETRY(G1)
    & FOUNDSPACE(O2,O,X,Y,Z) & HASLEVEL(G1,N) & CHOICECOUNT(K)
    & GETRIDCHOICE(K,G1,I,O2,O,X,Y,Z)
    & GETRIDCHOICE(K,G1,J,O2,O,X2,Y2,Z2)
    & NOT( EXISTS(O4,X3,Y3,Z3,L) & GETRIDCHOICE(K,G1,L,O4,O,X3,Y3,Z3)
        & SATISFIES2(L,J,L ?>GREAT J) )
  -> GETRIDOF(G1,O) & NEGATE(1,3) & GETRIDCHOICE(K,G1,J+1,O2,O,X,Y,Z)
    & TRACING(TRACEPRINTM(<FOUNDSPACE,'DUPLICATED,OK,Y,Z>));
W15: "GET RID RETRY O" = GETRIDOF(G,O) & RETRY(G) & CHOICECOUNT(N)
    & GETRIDCHOICE(N,G,L,A,O,X,Y,Z) & SATISFIES(L,L EQ 3) & ISA(O2,W)
    & NOT SATISFIES(W,W MEMQ '(TABLE PYRAMID BOX)) & VNEQ(O2,O)
    & NOT( EXISTS(I,X2,Y2,Z2) & GETRIDCHOICE(N,G,I,O2,O,X2,Y2,Z2) )
    & NOT( EXISTS(R,P) & HASREL(O,R,O2,P) & SATISFIES(R,R EQ 'ON) )
    & HASSIZE(O,SX,SY,SZ) & HASSIZE(O2,SX2,SY2,SZ2)
    & SATISFIES2(SX,SX2,NOT(SX2 ?>LESS SX))
    & SATISFIES2(SY,SY2,NOT(SY2 ?>LESS SY))
    & NOT( EXISTS(O3,SX3,SY3,SZ3,W2) & ISA(O3,W2) & VNEQ(O3,O2)
        & SATISFIES(W2,NOT(W2 MEMQ '(TABLE PYRAMID BOX))) & VNEQ(O3,O)
        & SATISFIES2(O2,O3,O3 LEXORDER O2)
        & NOT( EXISTS(I,X2,Y2,Z2) & GETRIDCHOICE(N,G,I,O3,O,X2,Y2,Z2) )
        & NOT( EXISTS(R,P) & HASREL(O,R,O3,P) & SATISFIES(R,R EQ 'ON) )
        & HASSIZE(O3,SX3,SY3,SZ3)
        & SATISFIES2(SX,SX3,NOT(SX3 ?>LESS SX))
        & SATISFIES2(SY,SY3,NOT(SY3 ?>LESS SY)) )
    % THAT MAKES CHOICE THE UNIQUE, LEXORDER'ST OBJECT %
  -> FINDSPACE(O2,O,SX,SY,SZ) & GETRIDPUT(G,O,O2) & NEGATE(1)
    & TRACING(TRACEPRINTM(<'TRYING,'ON,O2>));
W16: "GET RID EXH" = GETRIDOF(G,O) & RETRY(G) & CHOICECOUNT(N)
    & GETRIDCHOICE(N,G,L,A,O,X,Y,Z) & SATISFIES(L,L EQ 3)
    & HASSIZE(O,SX,SY,SZ)
    & NOT( EXISTS(O2,SX2,SY2,SZ2,W) & ISA(O2,W) & VNEQ(O2,O)
        & SATISFIES(W,NOT(W MEMQ '(PYRAMID TABLE BOX)))
        & NOT( EXISTS(I,X2,Y2,Z2) & GETRIDCHOICE(N,G,I,O2,O,X2,Y2,Z2) )
        & NOT( EXISTS(R,P) & HASREL(O,R,O2,P) & SATISFIES(R,R EQ 'ON) )
        & HASSIZE(O2,SX2,SY2,SZ2)
        & SATISFIES2(SX,SX2,NOT(SX2 ?>LESS SX))
        & SATISFIES2(SY,SY2,NOT(SY2 ?>LESS SY)) )
    % IE, NO OBJECTS ELIGIBLE AS CHOICES %
    & CHOICETIME(N,M) & HASLEVEL(G,K)
  -> ERSGETRIDCHOICES(N,G) & BACKUP(M-1) & CHOICECOUNT(N-1)
    & NEGATE(ALL,-6) & TRACING(TRACEPRINTM(<G,'EXHAUSTED>));
W16E: "ERS GETRID" = ERSGETRIDCHOICES(N,G) & GETRIDCHOICE(N,G,A,B,C,D,E,F)
  -> NEGATE(ALL);
W17: "FAIL GETRID FND 1" = FAILLOCATE(O) & GETRIDPUT(G,O,O2) & NOT RETRY(G)
    & CHOICECOUNT(K) & EVENTTIME(M)
  -> GETRIDOF(G,O) & RETRY(G) & CHOICECOUNT(K+1) & CHOICETIME(K+1,M)
    & GETRIDCHOICE(K+1,G,3,O2,O,O,O,O) & NEGATE(ALL,-5);
W18: "FAIL GETRID FND 2" = FAILLOCATE(O) & GETRIDPUT(G,O,O2) & RETRY(G)
    & CHOICECOUNT(K)
    & NOT( EXISTS(I,O3,X3,Y3,Z3) & GETRIDCHOICE(K,G,I,O3,O,X3,Y3,Z3)
        & SATISFIES(I,I EQ 3) )
  -> GETRIDOF(G,O) & NEGATE(1,2) & GETRIDCHOICE(K,G,3,O2,O,O,O,O);
W19: "FAIL GETRID FND C" = FAILLOCATE(O) & GETRIDPUT(G,O,O2) & RETRY(G)
    & CHOICECOUNT(K) & GETRIDCHOICE(K,G,I,O3,O,X,Y,Z)
    & SATISFIES(I,I ?>GREAT ?)
    & NOT( EXISTS(J,O4,X4,Y4,Z4) & GETRIDCHOICE(K,G,J,O4,O,X4,Y4,Z4)
        & SATISFIES2(I,J,J ?>GREAT I) )
  -> GETRIDOF(G,O) & NEGATE(1,2) & GETRIDCHOICE(K,G,I+1,O2,O,O,O,O);

    % PUT ON %

W20: "PUT ON SET" = PUTON(GT,O1,O2) & PUTON(GT,O3,O2) & VNEQ(O3,O1)
    & NOT( EXISTS(O4) & PUTON(GT,O4,O2) & VNEQ(O4,O1)
        & SATISFIES2(O4,O1,O4 LEXORDER O1) )
    & NOT( EXISTS(O4) & PUTON(GT,O4,O2) & VNEQ(O4,O1) & VNEQ(O4,O3)
        & SATISFIES3(O1,O3,O4,O1 LEXORDER O4 & O4 LEXORDER O3) )
        % MAKE FIRING UNIQUE %
    & CHOICECOUNT(K) & EVENTTIME(M)
    -> EXISTS(S) & PUTONSETO(S) & CHOICECOUNT(K+1) & CHOICETIME(K+1,M)
       & PUTONSETCHOICE(K+1,GT,S,02) & NEGATE(6);
W21: "PUT ON COLL" : PUTONSETO(S) & PUTON(GT,01,07)
    -> PUTONSET(GT,S,07) & INSET(01,S) & NEGATE(1,2);
W22: "PUT SEL" : PUTONSET(GT,S,01) & INSET(0,S) & NOT TRIEDPUT(0,S)
       & HASSIZE(0,SX,SY,SZ)
       & NOT( EXISTS(02,SX2,SY2,SZ2) & INSET(02,S) & NOT TRIEDPUT(02,S)
             & HASSIZE(07,SX2,SY2,SZ2)
             & SATISFIES3(SX,SY,SX2,SX2 + SY2 %GREAT SX + SY) )
             % PUT ON BIGGEST FIRST %
       & NOT( EXISTS(02,SX2,SY2,SZ2) & INSET(02,S) & NOT TRIEDPUT(02,S)
             & HASSIZE(07,SX2,SY2,SZ2)
             & VNEQ(02,0) & SATISFIES3(SX,SY,SX2,SX2 + SY2 = SX + SY)
             & SATISFIES2(02,0,02 LEXORDER 0) )
       & HASLEVEL(GT,N) & EVENTTIME(M)
    -> EXISTS(G) & PUTON1(G,0,01) & TRIEDPUT(0,S) & NEXT(G,<PUTONSET,GT,S,01>)
       & NEXTF(G,<FAILPUTONSET,GT,S,01>)
       & TRACING(TRACEPRINTM(<'DOING,GT,'PUTON,'SET,S,'0 1>)) & HASLEVEL(G,N+1)
       & TRACING(TRACEPRINTG(<G,'PUTON,0,'ONTO,01>,N+1)) & NEGATE(1,8)
       & NOCLEAR(GT) & UNEVENT(M,<ERSTRIEDPUT,0,S>) & EVENTTIME(M+1);
W22B: "BACK UP TRIED" : ERSTRIEDPUT(0,S) & EVENTTIME(M)
    -> NOT TRIEDPUT(0,S) & NEGATE(ALL) & EVENTTIME(M+1)
       & UNEVENT(M,<TRIEDPUT,0,S>);
W22S: "PUT ALL" : PUTONSET(GT,S,01)
       & NOT( EXISTS(0) & INSET(0,S) & NOT TRIEDPUT(0,S) )
    -> SUCCEED(GT) & NEGATE(1) & NOT NOCLEAR(GT);
W23: "PUT ON 1" : PUTON(GT,01,07)
       & NOT( EXISTS(03) & PUTON(GT,03,07) & VNEQ(03,01) )
    -> PUTON1(GT,01,07) & NEGATE(1) & NEXTF(GT,<FAILPUTON1,GT,01,02>)
       & TRACING(TRACEPRINTM(<'STARTING,GT,'PUTON,01,'ONTO,02>));
W23B: "PUT ON FAIL SET" : FAILPUTONSET(GT,S,0) & CHOICECOUNT(N)
    -> BACKUP(N) & NEGATE(1);
W23F: "PUT ON FAIL ALL" : FAILPUTONSETALL(GT,S,0) & ISA(0,W)
       & NOT SATISFIES(W,W EQ 'BOX) & HASLEVEL(GT,N)
    -> EXISTS(G) & CLEAROFF(G,0) & NEXT(G,<PACK,GT,S,0>) & HASLEVEL(G,N+1)
       & TRACING(TRACEPRINTG(<G,'CLEAROFF,0>,N+1)) & NEGATE(1);

       % PUT ON SINGLE OBJECT %

W24: "PUT ON" : PUTON1(GT,01,07) & HASSIZE(01,SX,SY,SZ) & HASLEVEL(GT,N)
       & HASSIZE(02,SX2,SY2,SZ2)
       & SATISFIES3(SX,SX2,SY,NOT(SX %GREAT SX2) & NOT(SY %GREAT SY2))
       & NOT( EXISTS(K,J,X3,Y3,Z3) & PUTON1CHOICE(K,GT,J,01,02,X3,Y3,Z3)
             & SATISFIES(J,J EQ 3) )
    -> EXISTS(G) & CLEAROFF(G,01) & NEXT(G,<FINDSPACE,02,01,SX,SY,SZ>)
       & HASLEVEL(G,N+1) & PUTONPUT(GT,01,07)
       & TRACING(TRACEPRINTG(<G,'CLEAROFF,01>,N+1)) & NEGATE(1);
W24F: "PUT ON OVER" : PUTON1(GT,01,02) & HASSIZE(01,SX,SY,SZ) & HASLEVEL(GT,N)
       & HASSIZE(02,SX2,SY2,SZ2)
       & NOT SATISFIES3(SX,SX2,SY,NOT(SX %GREAT SX2) & NOT(SY %GREAT SY2))
    -> FAIL(GT) & NEGATE(1)
       & TRACING(TRACEPRINTM(<01,'OVER,'SIZE,'OF,02>));
W25: "PUT ACT" : PUTONPUT(G1,01,07) & NOT RETRY(G1) & FOUNDSPACE(02,01,X,Y,Z)
       & HASLEVEL(G1,N) & CHOICECOUNT(K) & EVENTTIME(M)
    -> EXISTS(G) & PUT(G,01,X,Y,Z) & HASSUPERGOAL(G,G1) & HASLEVEL(G,N+1)
       & CHOICECOUNT(K+1) & CHOICETIME(K+1,M)
       & PUTON1CHOICE(K+1,G1,1,01,02,X,Y,Z)
       & NEGATE(1,3,5) & TRACING(TRACEPRINTG(<G,'PUT,01,<X,Y,Z>>,N+1));
W26: "PUT ACT RE" : PUTONPUT(G1,01,02) & RETRY(G1) & FOUNDSPACE(02,01,X,Y,Z)
       & CHOICECOUNT(K)
       & NOT( EXISTS(J) & PUTON1CHOICE(K,G1,J,01,02,X,Y,Z) )
       & PUTON1CHOICE(K,G1,J,01,07,X2,Y2,Z2)
       & NOT( EXISTS(J,X3,Y3,Z3) & PUTON1CHOICE(K,G1,J,01,02,X3,Y3,Z3)
             & SATISFIES2(J,I,J %GREAT I) )
       & HASLEVEL(G1,N)
    -> EXISTS(G) & PUT(G,01,X,Y,Z) & HASSUPERGOAL(G,G1) & HASLEVEL(G,N+1)
       & PUTON1CHOICE(K+1,G1,J+1,01,C7,X,Y,Z)
       & NEGATE(1,3) & TRACING(TRACEPRINTG(<G,'PUT,01,<X,Y,Z>>,N+1));
W26D: "PUT FND DUPL" : PUTONPUT(G1,01,02) & RETRY(G1) & FOUNDSPACE(02,01,X,Y,Z)
       & CHOICECOUNT(K) & PUTON1CHOICE(K,G1,I,01,07,X,Y,Z)
       & NOT( EXISTS(L) & PUTON1CHOICE(K,G1,L,01,02,X,Y,Z)
             & SATISFIES2(L,I,L %LESS I) )
       & PUTON1CHOICE(K,G1,J,01,07,X2,Y2,Z2)
       & NOT( EXISTS(L,X3,Y3,Z3) & PUTON1CHOICE(K,G1,L,01,02,X3,Y3,Z3)
             & SATISFIES2(L,J,L %GREAT J) )
    -> PUTON1(G1,01,07) & PUTON1CHOICE(K,G1,J+1,01,07,X,Y,Z)
       & NEGATE(1,3) & TRACING(TRACEPRINTM(<FOUNDSPACE,'DUPLICATED,<X,Y,Z>>));
W26X: "PUTON1 EXH" : PUTON1(G,01,07) & RETRY(G)
       & PUTON1CHOICE(K,G,I,01,07,X,Y,Z) & SATISFIES(I,I EQ 3) & CHOICECOUNT(K)
       & CHOICETIME(K,M) & HASLEVEL(G,L)
    -> ERSPUTON1CHOICES(K,G) & BACKUP(K-1) & CHOICECOUNT(K-1)
       & NEGATE(ALL) & TRACING(TRACEPRINTM(<G,'EXHAUSTED>));
W26Z: "ERS PUTON1" : ERSPUTON1CHOICES(K,G) & PUTON1CHOICE(K,G,L,01,02,X,Y,Z)
```

```
    -> NEGATE(ALL);
W27F: "LOCATE FAIL" : FAILLOCATE(0) & PUTONPUT(G,0,02)
    -> FAIL(G) & TRACING(TRACEPRINTM(<'NO,'SPACE,'TO,'PUTON,0,02>))
       & NEGATE(1,2);
W27M: "FAIL & MAKE" : FAILPUTON1(GT,01,02) & ISA(02,W)
       & NOT SATISFIES(W,W EQ 'BOX) & HASSIZE(01,SX,SY,SZ)
       & HASLEVEL(GT,N) & NOT CLEARTOP(02) & HASSIZE(02,SX2,SY2,SZ2)
       & SATISFIES3(SX2,SY2,SZ2,NOT(SX %GREAT SX2) & NOT(SY %GREAT SY2))
    -> EXISTS(G) & MAKESPACE(G,02,01,SX,SY,SZ) & HASLEVEL(G,N+1)
       & NEXT(G,<PUTONPUT,GT,01,02>)
       & TRACING(TRACEPRINTG(<G,'MAKESPACE,'FOR,01,'ON,02>,N+1)) & NEGATE(1);
W27O: "FAIL OVER" : FAILPUTON1(GT,01,02) & HASSIZE(01,SX1,SY1,SZ1)
       & HASSIZE(02,SX2,SY2,SZ2)
       & NOT SATISFIES3(SX1,SY1,SX2,
            NOT(SX1 %GREAT SX2) & NOT(SY1 %GREAT SY2))
    -> FAIL(GT) & NEGATE(1);
W27P: "FAIL CLEAR" : FAILPUTON1(GT,01,02) & CLEARTOP(02)
    -> FAIL(GT) & NEGATE(1);

END;

                    ......................

EXPR WBLOW3(): BEGIN                         % PAGE 5 %

       % BACK UP TO PREVIOUS CHOICE %

W30: "BACK SUB" : BACKUP(N) & EVENTTIME(K) & NOT( EXISTS(U) & UNEVENT(K,U) )
       & CHOICETIME(N,M) & SATISFIES2(K,M,NOT(K %LESS M))
    -> BACKUP(N) & EVENTTIME(K-1) & NEGATE(2);
W31: "BACK UP" : BACKUP(N) & EVENTTIME(K) & UNEVENT(K,U) & CHOICETIME(N,M)
       & SATISFIES2(K,M,NOT(K %LESS M))
    -> DELAYEXPND(MAKEINSTL(U)) & ERSUNEVENT(K,M) & NEGATE(1,3);
       % FOR MAKEINSTL, SEE W0 %
W32: "ERS UN" : ERSUNEVENT(K,M) & EVENTTIME(L) & SATISFIES2(L,K,L EQ K+1)
       & UNEVENT(K,U)
    -> BACKUP(N) & EVENTTIME(L-2) & NEGATE(ALL);
W33: "BACK GETRIDOF" : BACKUP(N) & CHOICETIME(N,M) & EVENTTIME(K)
       & SATISFIES2(K,M,K EQ M-1) & GETRIDCHOICE(N,G,J,02,0,X,Y,Z)
       & SATISFIES(J,J EQ 1) & HASLEVEL(G,L)
    -> GETRIDOF(G,0) & RETRY(G) & TRACING(TRACEPRINTG(<G,'RETRY,'GETRIDOF,0>,L))
       & EVENTTIME(M) & NEGATE(1,3);
W34: "BACK PUTON" : BACKUP(N) & CHOICETIME(N,M) & EVENTTIME(K)
       & SATISFIES2(K,M,K EQ M-1) & PUTON1CHOICE(N,G,J,01,02,X,Y,Z)
       & SATISFIES(J,J EQ 1) & HASLEVEL(G,L)
    -> PUTON1(G,01,07) & RETRY(G) & EVENTTIME(M) & NEGATE(1,3)
       & TRACING(TRACEPRINTG(<G,'RETRY,'PUTON1,01,02>,L));
W35: "BACK PACK" : BACKUP(N) & CHOICETIME(N,M) & EVENTTIME(K)
       & SATISFIES2(K,M,K EQ M-1) & PACKCHOICE(N,G,J,01,02,X,Y,Z)
       & SATISFIES(J,J EQ 1) & HASLEVEL(G,L) & INSET(01,S)
    -> PACK(G,S,07) & RETRY(G) & EVENTTIME(M) & NEGATE(1,3)
       & TRACING(TRACEPRINTG(<G,'RETRY,'PACK,S,02>,L));
W36: "BACK PUT ALL" : BACKUP(N) & CHOICETIME(N,M) & EVENTTIME(K)
       & SATISFIES2(K,M,K EQ M-1) & PUTONSETCHOICE(N,G,S,0)
       & HASLEVEL(G,L)
    -> FAILPUTONSETALL(G,S,0) & EVENTTIME(M) & NEGATE(1,3,5)
       & TRACING(TRACEPRINTG(<G,'RETRY,'WITH,'PACK>,L));

       % PUT IN %

       % PUT IN COMES FROM NL FRONT END AS PUTON %

W38: "PUTON-IN FAIL" : FAILPUTONSETALL(G1,S,07) & ISA(02,W)
       & SATISFIES(W,W EQ 'BOX) & HASLEVEL(G1,N)
    -> EXISTS(G) & ADDINSET('IN,07,S) & CLEAROFF(G,02)
       & NEXT(G,<PACK,G1,S,07>) & HASLEVEL(G,N+1)
       & TRACING(TRACEPRINTG(<G,'CLEAROFF,02>,N+1)) & NEGATE(1);
W38I: "PUTON-IN FAIL 1" : FAILPUTON1(G1,01,02) & ISA(02,W)
       & SATISFIES(W,W EQ 'BOX) & HASLEVEL(G1,N) & NOT CLEARTOP(02)
       & HASSIZE(01,SX1,SY1,SZ1) & HASSIZE(02,SX2,SY2,SZ2)
       & SATISFIES3(SX1,SX2,SY1,NOT(SX1 %GREAT SX2) & NOT(SY1 %GREAT SY2))
    -> EXISTS(G,S) & ADDINSET('IN,02,S) & CLEAROFF(G,02) & INSET(01,S)
       & NEXT(G,<PACK,G1,S,02>) & HASLEVEL(G,N+1)
       & TRACING(TRACEPRINTG(<G,'CLEAROFF,02>,N+1)) & NEGATE(1);
W39: "PACK IN COLL" : ADDINSET(R,0,S) & HASINDREL(02,R,0) & NOT INSET(02,S)
    -> INSET(02,S) & NEGATE(1);

       % STACK UP %

W60: "STACK UP START" : STACKUP(GT,0)
       & NOT( EXISTS(03) & STACKUP(GT,03) & VNEQ(03,0)
            & SATISFIES2(03,0,03 LEXORDER 0) )
      & ISA(02,W) & SATISFIES(W,W EQ 'TABLE)
   => EXISTS(S) & STACKSET(S) & INSET(0,S) & INSET(02,S) & TRIEDSTACK(02,S)
      & TRACING(TRACEPRINTM(<'STARTING,GT,'STACKUP>)) & NEGATE(1);
W41: "STACK SET" = STACKSET(S) & STACKUP(GT,0)
   => STACKUPSET(GT,S) & INSET(0,S) & NEGATE(ALL);
W42: "STACK PUT ON B" = STACKUPSET(GT,S) & INSET(0,S) & TRIEDSTACK(0,S)
      & NOT( EXISTS(02,P,R) & INSET(02,S) & HASREL(02,R,D,P)
            & SATISFIES(R,R EQ 'ON) & TRIEDSTACK(02,S) )
      & INSET(01,S) & NOT TRIEDSTACK(01,S) & ISA(01,W)
      & SATISFIES(W,W EQ 'BLOCK)
      & NOT( EXISTS(R,P) & HASREL(01,R,D,P) & SATISFIES(R,R EQ 'ON) )
      & HASSIZE(01,SX1,SY1,SZ1)
      & NOT( EXISTS(02,SX2,SY2,SZ2) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & ISA(02,W) & HASSIZE(02,SX2,SY2,SZ2)
            & SATISFIES3(SX2,SY2,SX1,SX2 * SY2 ?>GREAT SX1 * SY1) )
      & NOT( EXISTS(02,SX2,SY2,SZ2) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & ISA(02,W) & HASSIZE(02,SX2,SY2,SZ2)
            & SATISFIES3(SX2,SY2,SX1,SX2 * SY2 = SX1 * SY1)
            & SATISFIES2(02,01,02 LEXORDER 01) )
      & NOT( EXISTS(02,SX2,SY2,SZ2,R,P) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & ISA(02,W) & HASSIZE(02,SX2,SY2,SZ2)
            & SATISFIES3(SX2,SY2,SX1,SX2 * SY2 = SX1 * SY1)
            & HASREL(02,R,D,P) )
      & HASLEVEL(GT,N) & EVENTTIME(M)
   => EXISTS(G) & PUTON1(G,01,D) & NEXT(G,<'STACKUPSET,GT,S>) & HASLEVEL(G,N+1)
      & TRACING(TRACEPRINTG(<G,'PUTON1,01,'ONTO,D>,N+1)) & NEGATE(1,15)
      & NEXTF(G,<'FAILPUTONSTACK,GT,01,D,S>) & EVENTTIME(M+1)
      & TRIEDSTACK(01,S) & UNEVENT(M,<'ERSTRIEDSTACK,01,S>);
W42B: "BACK UP STACK" = ERSTRIEDSTACK(01,S) & EVENTTIME(M)
   => NOT TRIEDSTACK(01,S) & NEGATE(ALL) & EVENTTIME(M+1)
      & UNEVENT(M,<'TRIEDSTACK,01,S>);
W43: "STACK ON+" = STACKUPSET(GT,S) & INSET(0,S) & TRIEDSTACK(0,S)
      & INSET(01,S) & NOT TRIEDSTACK(01,S) & ISA(01,W)
      & SATISFIES(W,W EQ 'BLOCK) & HASREL(01,R,D,S) & SATISFIES(R,R EQ 'ON)
      & HASSIZE(01,SX1,SY1,SZ1)
      & NOT( EXISTS(02,SX2,SY2,SZ2) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & ISA(02,W) & HASSIZE(02,SX2,SY2,SZ2)
            & SATISFIES3(SX2,SY2,SX1,SX2 * SY2 ?>GREAT SX1 * SY1) )
      & EVENTTIME(M)
   => STACKUPSET(GT,S) & NEGATE(1,17)
      & TRACING(TRACEPRINTM(<'ALREADY,01,'ON,D>))
      & EVENTTIME(M+1) & TRIEDSTACK(01,S) & UNEVENT(M,<'ERSTRIEDSTACK,01,S>);
W44: "FAIL PUTON" = FAILPUTONSTACK(GT,01,0,S)
   => STACKUPSET(GT,S) & NEGATE(1)
      & TRACING(TRACEPRINTM(<'FAILED,'PUTON,'ONTO,D,'BUT,'PROCEED,'ANYWAY>));

W45: "STACK PUTON P" = STACKUPSET(GT,S) & INSET(0,S) & TRIEDSTACK(0,S)
      & NOT( EXISTS(02,P,R) & INSET(02,S) & HASREL(02,R,D,P)
            & SATISFIES(R,R EQ 'ON) & TRIEDSTACK(02,S) )
      & INSET(01,S) & NOT TRIEDSTACK(01,S) & ISA(01,W)
      & SATISFIES(W,W EQ 'PYRAMID) & NOT ISA(0,W)
      & NOT( EXISTS(R,P) & HASREL(01,R,D,P) & SATISFIES(R,R EQ 'ON) )
      & NOT( EXISTS(02,W2) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & ISA(02,W2) & SATISFIES(W2,W2 EQ 'BLOCK) )
      & NOT( EXISTS(02,R,P) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & ISA(02,W) & HASREL(02,R,D,P)
            & SATISFIES(R,R EQ 'ON) )
      & HASSIZE(01,SX1,SY1,SZ1)
      & NOT( EXISTS(02,SX2,SY2,SZ2) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & ISA(02,W) & VNEQ(02,01) & HASSIZE(02,SX2,SY2,SZ2)
            & SATISFIES3(SX2,SY2,SX1,SX2 * SY2 ?>GREAT SX1 * SY1) )
      & NOT( EXISTS(02,SX2,SY2,SZ2) & INSET(02,S) & ISA(02,W) & VNEQ(02,01)
            & HASSIZE(02,SX2,SY2,SZ2)
            & SATISFIES3(SX2,SY2,SX1,SX2 * SY2 = SX1 * SY1)
            & SATISFIES2(02,01,02 LEXORDER 01) )
      & HASLEVEL(GT,N) & EVENTTIME(M)
   => EXISTS(G) & PUTON1(G,01,0) & NEXT(G,<'STACKUPSET,GT,S>) & HASLEVEL(G,N+1)
      & TRACING(TRACEPRINTG(<G,'PUTON1,01,'ONTO,D>,N+1)) & NEGATE(1,17)
      & NEXTF(G,<'FAILPUTONSTACK,GT,01,D,S>) & EVENTTIME(M+1)
      & TRIEDSTACK(01,S) & UNEVENT(M,<'ERSTRIEDSTACK,01,S>);
W46: "STACK P ON+" = STACKUPSET(GT,S) & INSET(0,S) & TRIEDSTACK(0,S)
      & INSET(01,S) & NOT TRIEDSTACK(01,S) & ISA(01,W)
      & SATISFIES(W,W EQ 'PYRAMID) & HASREL(01,R,D,S) & SATISFIES(R,R EQ 'ON)
      & NOT( EXISTS(02,W2) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & ISA(02,W2) & SATISFIES(W2,W2 EQ 'BLOCK) )
      & EVENTTIME(M)
   => STACKUPSET(GT,S) & NEGATE(1,1)
      & TRACING(TRACEPRINTM(<'ALREADY,01,'ON,D>)) & EVENTTIME(M+1)
      & TRIEDSTACK(01,S) & UNEVENT(M,<'ERSTRIEDSTACK,01,S>);
W47: "STACK SUCC" = STACKUPSET(GT,S)
      & NOT( EXISTS(0) & INSET(0,S) & NOT TRIEDSTACK(0,S) )
   => SUCCEED(GT) & NEGATE(1);
W48: "STACK SUCC" = STACKUPSET(GT,S) & INSET(0,S) & TRIEDSTACK(0,S) & ISA(0,W)
      & SATISFIES(W,W EQ 'PYRAMID) & INSET(01,S) & NOT TRIEDSTACK(01,S)
      & NOT( EXISTS(02) & INSET(02,S) & NOT TRIEDSTACK(02,S)
            & VNEQ(02,01) & SATISFIES2(02,01,02 LEXORDER 01) )
   => SUCCEED(GT) & NEGATE(1)
      & TRACING(TRACEPRINTM(<'CANT,'COMPLETE,'STACK,01,'...>));

      % PACK %

W51: "PACK SEL BIG" = PACK(G1,S,D) & INSET(02,S) & NOT TRIEDPACK(02,S)
      & NOT( EXISTS(K,J,X,Y,Z) & PACKCHOICE(K,G1,J,02,D,X,Y,Z)
            & SATISFIES(J,J EQ 3) )
      & HASSIZE(02,SX2,SY2,SZ2)
      & NOT( EXISTS(03,SX3,SY3,SZ3) & INSET(03,S) & NOT TRIEDPACK(03,S)
            & HASSIZE(03,SX3,SY3,SZ3)
            & SATISFIES3(SX2,SY2,SX3,SX3 * SY3 ?>GREAT SX2 * SY2) )
      & NOT( EXISTS(03,SX3,SY3,SZ3) & INSET(03,S) & NOT TRIEDPACK(03,S)
            & VNEQ(03,02) & HASSIZE(03,SX3,SY3,SZ3)
            & SATISFIES3(SX2,SY2,SX3,SX3 * SY3 = SX2 * SY2)
            & SATISFIES2(03,02,03 LEXORDER 02) )
      & EVENTTIME(M)
   => LOCATESPACE(0,02,SX2,SY2,SZ2) & USERESULT(0,02,SX2,SY2,'PACK)
      & PACKPUT(G1,S,02,D) & TRIEDPACK(02,S) & UNEVENT(M,<'ERSTRIEDPACK,02,S>)
      & EVENTTIME(M+1) & NEGATE(1,8);
W51S: "PACK SUC" = PACK(G,S,D)
      & NOT( EXISTS(02) & INSET(02,S) & NOT TRIEDPACK(02,S) )
   => SUCCEED(G) & NEGATE(1);
W52B: "BACK UP PACK" = ERSTRIEDPACK(0,S) & EVENTTIME(M)
   => NOT TRIEDPACK(0,S) & NEGATE(ALL) & EVENTTIME(M+1)
      & UNEVENT(M,<'TRIEDPACK,0,S>);

W53: "PACK PUT B" = PACKPUT(G1,S,02,D) & NOT RETRY(G1) & FOUNDSPACE(0,02,X,Y,Z)
      & ISA(02,W) & SATISFIES(W,W EQ 'BLOCK) & HASLEVEL(G1,N)
      & CHOICECOUNT(K) & EVENTTIME(M)
   => EXISTS(G) & PUT(G,02,X,Y,Z) & NEXT(G,<'PACKUPON,G1,S,02,D>)
      & CHOICECOUNT(K+1) & CHOICETIME(K+1,M-1)
      % M-1 BECAUSE EVENT OF THIS CHOICE IS DONE BY W51 %
      & PACKCHOICE(K+1,G1,1,02,D,X,Y,Z) & HASLEVEL(G,N+1)
      & TRACING(TRACEPRINTG(<G,'PUT,02,'OK,Y,Z>,N+1)) & NEGATE(1,3,7);
W53A: "PACK PUT B RE" = PACKPUT(G1,S,02,D) & RETRY(G1) & FOUNDSPACE(0,02,X,Y,Z)
      & ISA(02,W) & SATISFIES(W,W EQ 'BLOCK) & CHOICECOUNT(K)
      & NOT( EXISTS(J) & PACKCHOICE(K,G1,J,02,D,X,Y,Z) )
      & PACKCHOICE(K,G1,1,02,D,X2,Y2,Z2)
      & NOT( EXISTS(J,X3,Y3,Z3) & PACKCHOICE(K,G1,J,02,D,X3,Y3,Z3)
            & SATISFIES2(J,I,J ?>GREAT I) )
      & HASLEVEL(G1,N)
   => EXISTS(G) & PUT(G,02,X,Y,Z) & NEXT(G,<'PACKUPON,G1,S,02,D>)
      & PACKCHOICE(K,G1,I+1,02,D,X,Y,Z) & HASLEVEL(G,N+1)
      & TRACING(TRACEPRINTG(<G,'PUT,02,'OK,Y,Z>,N+1)) & NEGATE(1,3);
W53D: "PACK FND DUPL" = PACKPUT(G1,S,02,D) & RETRY(G1) & FOUNDSPACE(0,02,X,Y,Z)
      & CHOICECOUNT(K) & PACKCHOICE(K,G1,1,02,D,X,Y,Z)
      & PACKCHOICE(K,G1,J,02,D,X2,Y2,Z2)
      & NOT( EXISTS(L,X3,Y3,Z3) & PACKCHOICE(K,G1,L,02,D,X3,Y3,Z3)
            & SATISFIES2(L,J,L ?>GREAT J) )
   => PACK(G1,S,D) & PACKCHOICE(K,G1,J+1,02,D,X,Y,Z) & NEGATE(1,3)
      & TRACING(TRACEPRINTM(<'FOUNDSPACE,'DUPLICATED,'OK,Y,Z>));

W54: "PACK PUT P" = PACKPUT(G1,S,02,D) & NOT RETRY(G1) & FOUNDSPACE(0,02,X,Y,Z)
      & ISA(02,W) & SATISFIES(W,W EQ 'PYRAMID) & HASLEVEL(G1,N)
      & CHOICECOUNT(K) & EVENTTIME(M)
   => EXISTS(G) & PUT(G,02,X,Y,Z) & NEXT(G,<'PACK,G1,S,D>)
      & CHOICECOUNT(K+1) & CHOICETIME(K+1,M-1) % CF. W53 %
      & PACKCHOICE(K+1,G1,1,02,D,X,Y,Z) & HASLEVEL(G,N+1)
      & TRACING(TRACEPRINTG(<G,'PUT,02,'OK,Y,Z>,N+1)) & NEGATE(1,3,7);
W54A: "PACK PUT P RE" = PACKPUT(G1,S,02,D) & RETRY(G1) & FOUNDSPACE(0,02,X,Y,Z)
      & ISA(02,W) & SATISFIES(W,W EQ 'PYRAMID) & CHOICECOUNT(K)
      & NOT( EXISTS(J) & PACKCHOICE(K,G1,J,02,D,X,Y,Z) )
      & PACKCHOICE(K,G1,I,02,D,X2,Y2,Z2)
      & NOT( EXISTS(J,X3,Y3,Z3) & PACKCHOICE(K,G1,J,02,D,X3,Y3,Z3)
            & SATISFIES2(J,I,J ?>GREAT I) )
      & HASLEVEL(G1,N)
   => EXISTS(G) & PUT(G,02,X,Y,Z) & NEXT(G,<'PACK,G1,S,D>)
      & PACKCHOICE(K,G1,I+1,02,D,X,Y,Z) & HASLEVEL(G,N+1)
      & TRACING(TRACEPRINTG(<G,'PUT,02,'OK,Y,Z>,N+1)) & NEGATE(1,[illegible]);
   % W54D = W53D %
W54X: "PACK EXH" = PACK(G,S,D) & RETRY(G) & PACKCHOICE(K,G,I,02,D,X,Y,Z)
      & SATISFIES(I,I EQ 3) & CHOICECOUNT(K) & CHOICETIME(K,M) & HASLEVEL(G,L)
   => ERSPACKCHOICES(K,G) & BACKUP(K-1) & CHOICECOUNT(K-1) & NEGATE(ALL)
      & TRACING(TRACEPRINTM(<G,'EXHAUSTED>));
W54Z: "ERS PACK" = ERSPACKCHOICES(K,G) & PACKCHOICE(K,G,L,01,02,X,Y,Z)
   => NEGATE(ALL);

W58: "PACK FAIL SP" = PACKPUT(G1,S,02,D) & FAILLOCATE(02) & CHOICECOUNT(K)
```

```
    -> BACKUP(E) & NEGATE(1,2);

WS6: "PACK UPON P" : PACKUPON(G1,S,07,D) & INSET(03,S) & NOT TRIEDPACK(03,S)
        & ISA(03,W) & SATISFIES(W,W EQ 'PYRAMID) & HASSIZE(03,SX3,SY3,SZ3)
        & HASSIZE(07,SX7,SY7,SZ7)
        & SATISFIES3(SX7,SY7,SX3,NOT(SX3 ?>GREAT SX7) & NOT(SY3 ?>GREAT SY7))
        & NOT( EXISTS(04,SX4,SY4,SZ4) & INSET(04,S) & VNEQ(04,03)
              & NOT TRIEDPACK(04,S) & ISA(04,W) & HASSIZE(04,SX4,SY4,SZ4)
              & SATISFIES3(SX2,SY2,SX4,NOT(SX4 ?>GREAT SX2)
                    & NOT(SY4 ?>GREAT SY2))
              & SATISFIES3(SX3,SY3,SX4,SX4 + SY4 ?>GREAT SX3 + SY3) )
        & NOT( EXISTS(04,SX4,SY4,SZ4) & INSET(04,S) & VNEQ(04,03)
              & NOT TRIEDPACK(04,S) & ISA(04,W) & HASSIZE(04,SX4,SY4,SZ4)
              & SATISFIES3(SX2,SY2,SX4,NOT(SX4 ?>GREAT SX2)
                    & NOT(SY4 ?>GREAT SY2))
              & SATISFIES3(SX3,SY3,SX4,SX4 + SY4 = SX3 + SY3)
              & SATISFIES2(04,03,04 LEXORDER 03) )
        & HASLEVEL(G1,N) & EVENTTIME(M)
     -> EXISTS(G) & PUTON1(G,03,07) & NEXT(G,'PACK,G1,S,D') & TRIEDPACK(03,S)
        & NEXTF(G,'FAILPACKUP,G1,03,D,S') & HASLEVEL(G,N+1)
        & UNEVENT(M,'ERSTRIEDPACK,03,S') & EVENTTIME(M+1)
        & TRACING(TRACEPRINTG('G,'PUTON,03,'ONTO,07',N+1)) & NEGATE(1,17);
WS7: "PACK UPON R" : PACKUPON(G1,S,07,0) & INSET(03,S) & NOT TRIEDPACK(03,S)
        & ISA(03,W) & SATISFIES(W,W EQ 'BLOCK) & HASSIZE(03,SX3,SY3,SZ3)
        & NOT( EXISTS(04,W2) & INSET(04,S) & NOT TRIEDPACK(04,S) & ISA(04,W2)
              & SATISFIES(W2,W2 EQ 'PYRAMID) )
        & HASSIZE(07,SX7,SY7,SZ7)
        & SATISFIES3(SX7,SY7,SX3,NOT(SX3 ?>GREAT SX7) & NOT(SY3 ?>GREAT SY7))
        & NOT( EXISTS(04,SX4,SY4,SZ4) & INSET(04,S) & VNEQ(04,03)
              & NOT TRIEDPACK(04,S) & ISA(04,W) & HASSIZE(04,SX4,SY4,SZ4)
              & SATISFIES3(SX7,SY7,SX4,NOT(SX4 ?>GREAT SX7)
                    & NOT(SY4 ?>GREAT SY7))
              & SATISFIES3(SX3,SY3,SY4,SX4 + SY4 ?>GREAT SX3 + SY3) )
        & NOT( EXISTS(04,SX4,SY4,SZ4) & INSET(04,S) & VNEQ(04,03)
              & NOT TRIEDPACK(04,S) & ISA(04,W) & HASSIZE(04,SX4,SY4,SZ4)
              & SATISFIES3(SX7,SY7,SX4,NOT(SX4 ?>GREAT SX7)
                    & NOT(SY4 ?>GREAT SY7))
              & SATISFIES3(SX3,SY3,SX4,SX4 + SY4 = SX3 + SY3)
              & SATISFIES2(04,03,04 LEXORDER 03) )
        & HASLEVEL(G1,N) & EVENTTIME(M)
     -> EXISTS(G) & PUTON1(G,03,07) & NEXT(G,'PACK,G1,S,D') & TRIEDPACK(03,S)
        & NEXTF(G,'FAILPACKUP,G1,03,D,S') & HASLEVEL(G,N+1)
        & UNEVENT(M,'ERSTRIEDPACK,03,S') & EVENTTIME(M+1) & NEGATE(1,12)
        & TRACING(TRACEPRINTG('G,'PUTON,03,'ONTO,07',N+1));
WS7F: "PACK UPON -" : PACKUPON(G1,S,07,0) & HASSIZE(07,SX7,SY7,SZ7)
        & NOT( EXISTS(03,SX3,SY3,SZ3) & INSET(03,S) & NOT TRIEDPACK(03,S)
              & HASSIZE(03,SX3,SY3,SZ3)
              & SATISFIES3(SX7,SY7,SX3,NOT(SX3 ?>GREAT SX7)
                          & NOT(SY3 ?>GREAT SY7)) )
     -> PACK(G1,S,D) & NEGATE(1);
WS8: "FAIL PACK UPON" : FAILPACKUP(G1,02,D,S) & TRIEDPACK(02,S) & EVENTTIME(M)
     -> PACK(G1,S,D) & NEGATE(ALL) & UNEVENT(M,'TRIEDPACK,02,S')
        & EVENTTIME(M+1);

END; END.
```

- -

```
BEGIN          % EXAMPLES FOR WBLOX %

EXPR WBLOY1(); BEGIN          PSMACRO(MIL,IM);

V1: TEST1(X) -> SAYQ(1,'(PUT THE SMALL RED BLOCK ON THE BLUE BLOCK));
V2: TEST2(X) -> SAYQ(2,'(WHAT IS BELOW THE SMALL RED BLOCK));
V3: TEST3(X)
   -> SAYQ(3,'(PUT THE GREEN BLOCK TO THE RIGHT OF THE LARGE RED BLOCK
           IN THE BOX));

END;

EXPR WBLOY2(); BEGIN          PSMACRO(MIL,IM);

V4: TEST4(X) -> SAYQ(4,'(PUT THE GREEN BLOCK ON THE BLOCK IN THE BOX));
V5: TEST5(X) -> SAYQ(5,'(WHAT IS IN THE BOX));
V6: TEST6(X) -> SAYQ(6,'(WHAT IS GREEN));
V7: TEST7(X)
   -> SAYQ(7,'(PUT THE GREEN PYRAMID AND THE RED PYRAMID ON THE BLUE BLOCK));

END;

EXPR WBLOY3(); BEGIN          PSMACRO(MIL,IM);

V8: TEST8(X) -> SAYQ(8,'(WHAT IS ON THE TABLE));
V9: TEST9(X)
   -> SAYQ(9,'(PUT THE LARGE RED BLOCK AND THE GREEN PYRAMID IN THE BOX));
      % WANT THAT TO FORCE PACK %
V10: TEST10(X) -> SAYQ(10,'(WHAT IS TO THE LEFT OF THE BOX));
V11: TEST11(X) -> SAYQ(11,'(WHAT IS IN FRONT OF THE BOX));


END;

EXPR WBLOY4(); BEGIN          PSMACRO(MIL,IM);

V12: TEST12(X)
   -> SAYQ(12,'(PUT A SMALL PYRAMID AND A SMALL PYRAMID AND A GREEN BLOCK
      AND THE SMALL RED BLOCK ON THE LARGE RED BLOCK));
      % THAT WILL FORCE PACK %
V13: TEST13(X) -> SAYQ(13,'(PUT THE BLUE BLOCK IN THE BOX));
V14: TEST14(X)
   -> ISA('BLOCK?-6,'BLOCK) & ISA('BLOCK?-7,'BLOCK)
      & ISA('BLOCK?-8,'BLOCK) & ISA('BLOCK?-9,'BLOCK)
      & LOCAT('BLOCK?-6,100,0,0) & LOCAT('BLOCK?-7,400,0,0)
      & LOCAT('BLOCK?-8,600,0,0) & LOCAT('BLOCK?-9,800,0,0)
      & HASREL('BLOCK?-6,'ON,'TABLE?-1,'POS)
      & HASREL('BLOCK?-7,'ON,'TABLE?-1,'POS)
      & HASREL('BLOCK?-8,'ON,'TABLE?-1,'POS)
      & HASREL('BLOCK?-9,'ON,'TABLE?-1,'POS)
      & HASSIZE('BLOCK?-6,200,200,200) & HASSIZE('BLOCK?-7,200,200,200)
      & HASSIZE('BLOCK?-8,200,200,200) & HASSIZE('BLOCK?-9,200,200,200)
      & HASAV('BLOCK?-6,'COLOR,'BLACK,'POS)
      & HASAV('BLOCK?-7,'COLOR,'BLACK,'POS)
      & HASAV('BLOCK?-8,'COLOR,'BLACK,'POS)
      & HASAV('BLOCK?-9,'COLOR,'BLACK,'POS)
      & HASAV('BLOCK?-6,'SIZE,'LARGE,'POS)
      & HASAV('BLOCK?-7,'SIZE,'LARGE,'POS)
      & HASAV('BLOCK?-8,'SIZE,'LARGE,'POS)
      & HASAV('BLOCK?-9,'SIZE,'LARGE,'POS)
      & CLEARTOP('BLOCK?-6) & CLEARTOP('BLOCK?-7)
      & CLEARTOP('BLOCK?-8) & CLEARTOP('BLOCK?-9);

END;

EXPR WBLOY5(); BEGIN          PSMACRO(MIL,IM);

V15: TEST15(X) -> SAYQ(15,'(PUT A BLACK BLOCK ON THE LARGE RED BLOCK));
      % ANOTHER FORM OF FAIL - WILL DO MAKESPACE %
V16: TEST16(X) -> SAYQ(16,'(PUT A LARGE GREEN BLOCK IN THE BOX));
      % HOPE TO FORCE CLEAROUT; ALSO AMBIGUOUS %
V17: TEST17(X) -> SAYQ(17,'(PICK A BLACK BLOCK UP));

END;

EXPR WBLOY6(); BEGIN          PSMACRO(MIL,IM);

V18A: TEST18A(X) -> SAYQ(18.0,'(PUT IT IN THE BOX));
V18B: TEST18B(X) -> SAYQ(18.5,'(PICK A BLACK BLOCK ON THE TABLE UP));
      % SIMPLY REPEAT TEST 18A TO TRY TO FORCE CLEAR-OUT OF BOX,
           WITH A BACKUP OF PACK, HOPEFULLY %

END;

EXPR WBLOY7(); BEGIN          PSMACRO(MIL,IM);

V19: TEST19(P) & LOCAT(H,X,Y,Z) & SATISFIES(H,H EQ 'HAND?-1)
   -> CLEARTOP('BLOCK?-A) & GRASPING('HAND?-1,'BLOCK?-A) % NO COLOR %
      & HASAV('BLOCK?-A,'SIZE,'LARGE,'POS) & HASSIZE('BLOCK?-A,200,200,100)
      & ISA('BLOCK?-A,'BLOCK) & LOCAT('BLOCK?-A,X-100,Y-125,Z-100)
      & SAYQ(19,'(STACK UP A LARGE RED BLOCK AND A SMALL BLOCK AND IT
           AND A SMALL PYRAMID AND A BLACK BLOCK
           AND A LARGE GREEN BLOCK AND A SMALL PYRAMID))
      & NEGATE(1);
V20: TEST20(P) & LOCAT(H,X,Y,Z) & SATISFIES(H,H EQ 'HAND?-1)
   -> CLEARTOP('BLOCK?-0) & GRASPING('HAND?-1,'BLOCK?-0) % NO COLOR %
      & HASAV('BLOCK?-0,'SIZE,'LARGE,'POS) & HASSIZE('BLOCK?-0,200,200,100)
      & ISA('BLOCK?-0,'BLOCK) & LOCAT('BLOCK?-0,X-150,Y-150,Z-100)
      & SAYQ(20,'(PUT IT DOWN)) & NEGATE(1);
V21: TEST21(X) -> SAYQ(21,'(PUT THE LARGE BLUE BLOCK AND
           THE LARGE PYRAMID ON THE TABLE));

END;

EXPR WBLOY8(); BEGIN          PSMACRO(MIL,IM);

V22: TEST22(P) & LOCAT(H,X,Y,Z) & SATISFIES(H,H EQ 'HAND?-1)
```

```
    -> CLEARTOP('PYRAMID?-B) & GRASPING('HAND?-1,'PYRAMID?-B) % NO COLOR %
      & HASAV('PYRAMID?-B,'SIZE,'LARGE,'POS)
      & HASSIZE('PYRAMID?-B,400,220,100)
      & ISA('PYRAMID?-B,'PYRAMID) & LOCAT('PYRAMID?-B,X-200,Y-110,Z-100)
      & SAYQ(22,'(PUT IT DOWN)) & NEGATE(1);
V23; TEST23(P) & LOCAT(H,X,Y,Z) & SATISFIES(H,H EQ 'HAND?-1)
    -> CLEARTOP('BLOCK?-C) & GRASPING('HAND?-1,'BLOCK?-C) % NO COLOR %
      & HASAV('BLOCK?-C,'SIZE,'LARGE,'POS) & HASSIZE('BLOCK?-C,400,220,100)
      & ISA('BLOCK?-C,'BLOCK) & LOCAT('BLOCK?-C,X-200,Y-110,Z-100)
      & SAYQ(23,'(PUT IT DOWN)) & NEGATE(1);
V24; TEST24(X) -> SAYQ(24,'(PICK UP THE LARGE RED BLOCK));
      % PICK UP BOTTOM OF STACK - FORCED GETRIDOF BACKUP %
      % THIS SHOULD BACK UP BECAUSE IT WILL TRY PLACING A SMALL
            BLOCK ON THE HUGE ONE (BLOCK-D, LEXORDER THE OTHERS)
            AND HAVE TO UNDO IT TO GET RID OF THE ALMOST-HUGE ONE
            (BLOCK-A) %
END;

END.
```

Appendix G. CROSS-REFERENCE OF WELDM PREDICATES

```
ADDAV
 LHSUSES M51
 RHSUSES M41 M42 M43 M44 M45 -M51
ADDINSET
 LHSUSES W39
 RHSUSES W38 W381 -W39
ANSPRED
 LHSUSES V35 V36 V37
 NESTEDL V46 V49
 RHSUSES M15 V48 V49
ANSPREDFIN
 LHSUSES V40 V44 V48
 NESTEDL V42 V46
 RHSUSES A14 V44 -V48
ANSPREDRED
 LHSUSES V49
 RHSUSES M16
ANSREL
 LHSUSES V30 V31 V32
 NESTEDL V49
 RHSUSES V42 V46
ANSRELINC
 LHSUSES V40 V42
 NESTEDL V46 V48 V49
 RHSUSES M11
ANSRELRED
 LHSUSES V44 V46
 NESTEDL V48
 RHSUSES M12 -V44
AVRESTR
 LHSUSES F27 F29
 RHSUSES A1 -F29
BACKUP
 LHSUSES W30 W31 W33 W34 W35 W36
 RHSUSES W16 W23B W26X W30 -W31 W32 -W33 -W34 -W35 -W36 W54X W56
CHAINREL
 LHSUSES B10J B10K B10L
 RHSUSES B10I B10J -B10J B10K -B10K -B10L
CHECKFAILFIT
 LHSUSES Q69
 RHSUSES Q67 -Q68 -Q69
CHECKPICKUP
 LHSUSES V51
 RHSUSES M81 -V51
CHECKPICKUP2
 LHSUSES V51A V51H V51O
 RHSUSES V51 -V51A -V51H
CHECKPUTDOWN
 LHSUSES V53
 RHSUSES M84 -V53
CHECKPUTDOWN2
 LHSUSES V53O V53L V53R
 RHSUSES V53 -V53O -V53L -V53R
CHECKPUTON
 LHSUSES V52
 RHSUSES M82 M83 -V52
CHECKPUTON2
 LHSUSES V52A V52F
 RHSUSES V52 -V52A -V52F
CHECKSTACKUP
 LHSUSES V54
 RHSUSES M86 -V54
CHECKSTACKUP2
 LHSUSES V54A V54F
 NESTEDL V54A
 RHSUSES V54 -V54A -V54F
CHOICECOUNT
 LHSUSES W12 W13 W14 W140 W15 W16 W17 W18 W19 W20 W23B W25 W26 W26D W26X W53
 W53A W53D W54 W54A W54X W55
 RHSUSES W12 -W12 W16 -W16 W17 -W17 W20 -W20 W25 -W25 W26X -W26X W53 -W53 W54
 -W54 W54X -W54X M89
CHOICETIME
 LHSUSES W16 W26X W30 W31 W33 W34 W35 W36 W54X
 RHSUSES W12 -W16 W17 W20 W25 -W26X W53 W54 -W54X
CLEAROFF
 LHSUSES W3 W4 W6
 RHSUSES Q45 -W3 -W4 -W6 W23F W24 W38 W381
CLEARTOP
 LHSUSES Q27 Q57 Q57F -Q61 W6 -W27M W27P -W381
 RHSUSES -Q21 Q23 -Q27
```

```
CONJBOUND
  LHSUSES B59I
  NESTEDL B59
  RHSUSES G45 -B59I
CONVIND
  LHSUSES F61 F62 F63 F64 F65 F66
  RHSUSES -F61 -F62 -F63 -F64 -F65 -F66 B10C
COPSIGN
  LHSUSES R11
  NESTEDL R12
  RHSUSES G31 -G31 G32 -G32 -R11
CUROBJ
  LHSUSES A1 A5 R11 R12 N1 N3 N5 N5A N31 N33 F41 F51 F53 B1 B18 B24 B33 B33I
  -B34 -B34I B38 B43 -B44 B48 B51 -B55 -B56 B58 M71 V10 V12 V14 V17 V19 V29
  NESTEDL N2 N6 B17 B27 B39 M87
  RHSUSES T66 G13 N1 -N1 N2 N5 -N5 N6 -N31 N41 N42 N43 N44 N45 B33 -B33 B33I
  -B33I B38 -B38 B43 -B43 B48 -B48 -B51 B55 B56 -B59I
CUROBJP
  LHSUSES F34 F34I B3 B19I B33 B33I B34 B34I B35 B36 B38 B39 B43 B44 B45 B46
  B48 B53 B55 B56 B59I M1 M2 M5 M11 M12 M15 M18 M51 M53 M61 M62 V48
  NESTEDL B17 B55 B57 M1 M2 M5
  RHSUSES T66 G13 N1 N2 N5 B1 B33 -B33 B33I -B33I B34 -B34 B34I -B34I -B38 B39
  -B39 B43 -B43 B44 -B44 -B48 -B53 -B59I
DEFDET
  LHSUSES N1 N2 N22 -N29
  RHSUSES G1 G2 G5 G51
DEFFND
  LHSUSES F5 F6
  RHSUSES N1 N2 -F5 -F6
DESCRAV
  LHSUSES D2 D3 D4 D11 D12
  RHSUSES D1 D2 -D2 D3 -D3 -D4 D11 -D11 D12 -D12
DESCRIBE
  LHSUSES D1
  RHSUSES V10 V14 V17 V18 V19 -D1
DESCRIBED
  LHSUSES -D11 -D12
  NESTEDL -D2 -D3 -D4 -D11 -D12
  RHSUSES D11 D12
DESCRNX
  LHSUSES D3 D4
  RHSUSES D1
DESCRPHRASE
  LHSUSES V15 D21 D22 D23 D25 D29
  NESTEDL V15
  RHSUSES -V15 D4
DETSEEN
  LHSUSES A19 -A25 -N1 -N2 N3 -N5 -N6
  RHSUSES N1 N2 N5 N6
ENDMARK
  LHSUSES S0 -S1 S4 T67 -E4 E6 A14 N80 M65 M66
  NESTEDL A25
  RHSUSES Y1
EQA
  LHSUSES G5 G51 G6 G7
  RHSUSES -G5 -G51 -G6 -G7
EQABOVE
  LHSUSES T87
  RHSUSES -T87
EQAND
  LHSUSES G45
  RHSUSES -G45
EQBEHIND
  LHSUSES T86
  RHSUSES -T86
EQBELOW
  LHSUSES T88
  RHSUSES -T88
EQBLACK
  LHSUSES T18
  RHSUSES -T18
EQBLOCK
  LHSUSES T44
  RHSUSES Y1 -T44
EQBLUE
  LHSUSES T13
  RHSUSES Y1 -T13
EQBOX
  LHSUSES T53
  RHSUSES -T53
EQDOWN
  LHSUSES T72
  RHSUSES -T72
EQFLOOR
  LHSUSES T50
  RHSUSES -T50
EQFRONT
  LHSUSES T83
  RHSUSES -T83
EQGRASP
  LHSUSES G43
  RHSUSES -G43
EQGREEN
  LHSUSES T10
  RHSUSES -T10
EQIN
  LHSUSES T31 T83
  RHSUSES -T31 -T83
EQIS
  LHSUSES T1 T2 T4
  RHSUSES -T1 -T4
EQIT
  LHSUSES T66 T67
EQLARGE
  LHSUSES T21
  RHSUSES -T21
EQLEFT
  LHSUSES T81
  RHSUSES -T81
EQMEDIUM
  LHSUSES T24
  RHSUSES -T24
EQNEAR
  LHSUSES T37
  RHSUSES -T37
EQNOT
  LHSUSES -T1 T2 T4
  RHSUSES -T4
EQOF
  LHSUSES T81 T82 T83
  RHSUSES -T81 -T82 -T83
EQON
  LHSUSES T34
  RHSUSES Y1 -T34
EQPICK
  LHSUSES G41
  RHSUSES -G41
EQPUT
  LHSUSES G44
  RHSUSES Y1 -G44
EQPYRAMID
  LHSUSES T41
  RHSUSES -T41
EQRED
  LHSUSES T7
  RHSUSES Y1 -T7
EQRIGHT
  LHSUSES T82
  RHSUSES -T82
EQSMALL
  LHSUSES T27
  RHSUSES Y1 -T27
EQSTACK
  LHSUSES G42
  RHSUSES -G42
EQTABLE
  LHSUSES T47
  RHSUSES -T47
EQTHAT
  LHSUSES T63
  RHSUSES -T63
EQTHE
  LHSUSES T81 T82 G1 G2
  RHSUSES Y1 -T81 -T82 -G1 -G2
EQTHERE
  LHSUSES G9 G10 G17 -G18
  RHSUSES -G9 -G10
EQTO
  LHSUSES T81 T82
  RHSUSES -T81 -T82
EQUNDER
  LHSUSES T39
  RHSUSES -T39
EQUP
  LHSUSES T71
  RHSUSES -T71
```

```
EQWHAT
  LHSUSES T87
  RHSUSES -T87
EQWHERE
  LHSUSES G21
  RHSUSES -G21
EQWHICH
  LHSUSES T80
  RHSUSES -T80
ERROR
  LHSUSES E2
  RHSUSES S7 T67 -E2 A25 R5 P9 N10U N29 F2 F6 F51 F53 B17 B27 B57 B58F M51 M60
  M62 M64A M65 M66 V40
ERRORS
  LHSUSES E4 E6 E8
  RHSUSES E2 E4 -E4 -E6 E8 -E8
ERRREF
  LHSUSES E8 B1 B3 B57 B58F
  NESTEDL N31 N33
  RHSUSES T66 -E8 G13 N31 N33 N41 N42 N43 N44 N48
ERSFINDNEARPAIR
  LHSUSES Q64E
  RHSUSES Q64A Q64B -Q64E
ERSFINDPOSS
  LHSUSES B58E
  RHSUSES B58C -B58E
ERSGETRIDCHOICES
  LHSUSES W16E
  RHSUSES W16 -W16E
ERSPACKCHOICES
  LHSUSES W54Z
  RHSUSLS W54X -W54Z
ERSPUTONICHOICES
  LHSUSES W26Z
  RHSUSES W26X -W26Z
ERSREMOHASREL
  LHSUSES Q29
  RHSUSES Q6 -Q29
ERSTRIEDPACK
  LHSUSES W52B
  RHSUSES -W52B
ERSTRIEDPUT
  LHSUSES W22B
  RHSUSES -W22B
ERSTRIEDSTACK
  LHSUSES W42B
  RHSUSES -W42B
ERSUNEVENT
  LHSUSES W32
  RHSUSES W31 -W32
EVENTTIME
  LHSUSES Q1 Q2 Q47 Q47U Q49 W12 W17 W20 W22 W22B W25 W30 W31 W32 W33 W34 W35
  W36 W42 W42B W43 W45 W46 W51 W52B W53 W54 W56 W57 W58
  RHSUSES Q1 -Q1 Q2 -Q2 Q47 -Q47 Q47U -Q47U Q49 -Q49 W22 -W22 W22B -W22B W30
  -W30 W32 -W32 W33 -W33 W34 -W34 W35 -W35 W36 -W36 W42 -W42 W42B -W42B W43
  -W43 W45 -W45 W46 -W46 W51 -W51 W52B -W52B W56 -W56 W57 -W57 W58 -W58 M89
EXPECTMOD
  LHSUSES T71 T72 F34 F34I M81 M82 M84 M84A M81F M88F
  NESTEDL M84 M85 M81 M88
  RHSUSES -T71 -T72 G41 G42 G44 -M81F -M88F
FAIL
  LHSUSES W0F W0G
  RHSUSES -W0F -W0G W24F W27F W27O W27P
FAILLOCATE
  LHSUSES Q85 W17 W18 W19 W27F W55
  RHSUSES Q57F Q63 Q64B -Q85 -W17 -W18 -W19 -W27F -W55
FAILPACKUP
  LHSUSES W58
  RHSUSES -W58
FAILPUTONI
  LHSUSES W27M W27O W27P W38I
  RHSUSES -W27M -W27O -W27P -W38I
FAILPUTONSET
  LHSUSES W23B
  RHSUSES -W23B
FAILPUTONSETALL
  LHSUSES W23F W38
  RHSUSES -W23F W38 -W38
FAILPUTONSTACK
  LHSUSES W44
  RHSUSES -W44
FINDAMBIGP
  LHSUSES B43 B44 B45 B46
  RHSUSES B41 -B43 -B44 B45 -B45 -B46
FINDAMBIGR
  LHSUSES B33 B33I B34 B34I B35 B36
  RHSUSES B31 B31I -B33 -B33I -B34 -B34I B35 -B35 -B36
FINDHIGHX
  LHSUSES Q70 Q73
  RHSUSES Q68 -Q73
FINDHIGHY
  LHSUSES Q71 Q73
  RHSUSES Q68 -Q73
FINDLOWPAIR
  LHSUSES Q62 Q63 Q64 Q64A Q64B
  RHSUSES Q61 -Q62 -Q63 Q64A -Q64A -Q64B Q69
FINDLOWX
  LHSUSES Q65 Q68 Q69
  RHSUSES Q62 -Q68 -Q69
FINDLOWY
  LHSUSES Q66 Q68 Q69
  RHSUSES Q62 -Q68 -Q69
FINDNEARPAIR
  LHSUSES Q64A Q64B Q64E
  NESTEDL Q64A Q64B
  RHSUSES Q64 -Q64A -Q64B -Q64E
FINDPOSS
  LHSUSES F13 F15 F21 F27 F31 F32 F32C F34 F34I F35 F61 F62 F63 F64 F65 F66 B13
   B13I B17 B19C B19I B23 B27 B31 B31I B33 B33I B34 B34I B36 B41 B43 B44 B46
  B56 B57 B58 B58C B58E V14
  NESTEDL F11 F13 F15 F34 F34I B17 B19C B27 B35 B36 B45 B46 B55 B56 B58C B58F
  M1 M2 M5 M12 M16
  RHSUSES F1 F5 -F13 -F21 -F27 -F31 -F32 -F32C -F35 -B19I -B58E
FINDSPACE
  LHSUSES Q51 Q52 Q53 Q54
  RHSUSES -Q51 -Q52 -Q53 -Q54 Q83 W11 W13 W15
FOUNDHIGHPAIR
  LHSUSES Q73
  RHSUSES Q72 -Q73
FOUNDHIGHPAIRO
  LHSUSES Q72
  RHSUSES Q69 -Q72
FOUNDSPACE
  LHSUSES Q84 W12 W14 W14O W25 W26 W26D W53 W53A W53D W54 W54A
  RHSUSES Q76 Q77 Q78 -W12 -W14 -W14O -W25 -W26 -W26D -W53 -W53A -W53D -W54
  -W54A
GETRIDCHOICE
  LHSUSES W14 W14O W15 W16 W16E W19 W23
  NESTEDL W13 W14 W14O W15 W16 W18 W19
  RHSUSES W12 W14 W14O -W16 -W16E W17 W18 W19
GETRIDOF
  LHSUSES W11 W13 W15 W16
  RHSUSES Q43 Q61 Q82 W3 W4 W10 -W11 -W13 W14O -W15 -W16 W17 W18 W19 W23
GETRIDPUT
  LHSUSES W12 W14 W14O W17 W18 W19
  RHSUSES W11 -W12 W13 -W14 -W14O W15 -W17 -W18 -W19
GRASP
  LHSUSES Q41 Q43 Q45
  RHSUSES Q31 -Q41 -Q43 -Q45 W1
GRASP1
  LHSUSES Q46
  RHSUSES -Q46
GRASP2
  LHSUSES Q47
  RHSUSES Q46 -Q47
GRASP3
  LHSUSES Q47U
  RHSUSES -Q47U
GRASPING
  LHSUSES Q2 Q2L Q41 Q43 Q49 T86 M88 M84 V51A V51D
  NESTEDL Q1 Q45 T67 M84F V51H
  RHSUSES Q47 Q47U -Q49
GROWTOFIT
  LHSUSES Q68
  RHSUSES Q67 -Q68 -Q69
GROWTOFITO
  LHSUSES Q67
  RHSUSES Q62 -Q67
GSD
  LHSUSES N13 M1 M2 M5 -M11 -M12 -M51 -M53 V2
  RHSUSES G2 G7
GSE
  LHSUSES -M11 -M12 V20
  RHSUSES G9
GS1
  LHSUSES G5I -G6 G45 F34 F34I B19I B58 B58I -M11 -M12 -M51 M81 M82 M83 M84
  M64A M65 M66 M71
  NESTEDL B17 B58
  RHSUSES G41 G42 G43 G44
GSQD
  LHSUSES A14 -M81 -M83
  NESTEDL A25
  RHSUSES G18
GSQE
  LHSUSES G5 -G6 G10 N9A -F23 M15 M16 -M81 -M83 V25
  RHSUSES G17
GSQW
  LHSUSES N18 -F33 B14 B24 -M11 -M12 V10 V12 V16
  NESTEDL B17 B27 B55 B57
  RHSUSES G13
GSQWR
  LHSUSES -M11 -M12 V17 V19
  RHSUSES G21
GTYPED
  LHSUSES G1 -G2 G6 -G7 -G9 -G13 -G17 -G18 -G21 -G41 -G42 -G43 -G44
  RHSUSES G2 G7 G9 G13 G17 G18 G21 G41 G42 G43 G44
HASAV
  LHSUSES E11 -F27 -F29 -F35 -B21 B23 B25 -B28 B29 B41 B43 B44 -B46 V35 V36 D11
   D12
  NESTEDL B21 B27 B46 M16 V37 D2 D3 D4 D11 D12
  RHSUSES N51 B21 M2 M5
HASINDREL
  LHSUSES W39 -F32 -F32C -F34 F34I -F34I B13I B15I -B18 B31I B33I B34I -B36
  NESTEDL B17 B19C B36
  RHSUSES F61 F62 F63 F64 F65 F66 B10J B10K B10L
HASLEVEL
  LHSUSES Q31 Q43 Q45 Q81 Q82 W0 W0F W0G W0S W0T W1 W3 W4 W10 W12 W14 W14D W16
  W22 W23F W24 W24F W25 W26 W26X W27M W33 W34 W35 W36 W38 W38I W42 W43 W53 W53A
   W54 W54A W54X W56 W57
  RHSUSES Q31 Q43 Q45 Q81 Q82 -W0T W1 W3 W4 W10 W12 W14 -W16 W22 W23F W24 W25
  W26 -W26X W27M W38 W38I W42 W45 W53 W53A W54 W54A -W54X W56 W57 M89
HASREL
  LHSUSES Q6 Q15 Q17 Q21 Q27 Q49 Q81 Q82 W3 W4 W43 W46 E12 -F31 F34 -F34 -F34I
  B10J B10K B10L -B11 B13 B15 -B18 B19 B31 B33 B34 -B36 V17 V18 V30 V31 V51D
  V52A V53D
  NESTEDL Q23 Q81 Q82 W3 W4 W15 W16 W42 W45 B11 B17 B19C B36 M12 V19 V32 V51A
  V52F V53R
  RHSUSES -Q6 Q7 Q21 -Q21 B11 M1
HASRELN
  LHSUSES B1 B3
  RHSUSES R11 R12
HASSIZE
  LHSUSES Q2 Q2L Q7 Q32 Q45 Q57 Q57F Q61 Q64 Q65 Q66 Q70 Q71 Q81 Q82 W3 W4 W11
  W13 W15 W16 W22 W24 W24F W27M W270 W38I W42 W43 W45 W51 W56 W57 W57F
  NESTEDL Q2 Q62 Q65 Q66 Q68 Q70 Q71 Q81 Q82 W3 W4 W15 W16 W22 W42 W43 W45 W51
  W56 W57 W57F
HASSUPERGOAL
  LHSUSES W0S
  NESTEDL W0T
  RHSUSES W10 W12 W14 W28 W28
HIGHX
  LHSUSES Q70 Q73
  RHSUSES Q68 Q70 -Q70 -Q73
HIGHY
  LHSUSES Q71 Q73
  RHSUSES Q68 Q71 -Q71 -Q73
IMPCHOICE
  LHSUSES -B58C
  NESTEDL -B58C -B58F
  RHSUSES B58C
IMPCHOOSE
  LHSUSES B58C B58F
  RHSUSES B19C B58 -B58C -B58F
IMPINDEF
  LHSUSES N3 B19C B56 B58
  NESTEDL B17 B56 B57
  RHSUSES G51 N3 -N3
IMPOBJ
  LHSUSES M66 M81 M82 M82P M83 M83P M84 M84F M85 M86
  NESTEDL M87
  RHSUSES B58I M71
IMPREL
  LHSUSES M62 M66 M71 M82 M82P M83 M83P M84 M84F
  NESTEDL M61 M63 M64 M64A M65 M85
  RHSUSES T72 G41 G42 G43 M81 M83 M84
IMPRESTR
  LHSUSES M63
  NESTEDL M64 M64A M65
  RHSUSES F34 F34I
IMPTYPE
  LHSUSES M81 M81F M82 M82P M83 M83P M84 M84F M85 M86 M86F M87
  RHSUSES G41 G42 G43 G44
INDEFDET
  LHSUSES N5 N8 N23 -N29
  RHSUSES G6 G7
INSET
  LHSUSES W22 W35 -W39 W42 W43 W45 W46 W48 W51 W56 W57
  NESTEDL W22 W22S W42 W43 W45 W46 W47 W48 W51 W51B W56 W57 W57F
  RHSUSES W21 W38I W38 W40 W41
INSTACK
  LHSUSES Q11 Q13 Q15 -Q15 V54A V54F -V54F
  NESTEDL Q13 Q17 V54A -V54A V54F
  RHSUSES -Q11 -Q13 Q15 Q17
ISA
  LHSUSES Q1 Q3 Q7 Q21 Q27 Q47 Q51 Q52 Q53 Q54 W1 W11 W13 W15 W23F W27M W38
  W38I W40 W42 W43 W45 -W45 W46 W48 W53 W53A W54 W54A W56 W57 E13 F1 F8 -F21
  -F23 B10K B10L M82 M82P M83 M83P D4
  NESTEDL Q17 W15 W16 W42 W43 W45 W46 W56 W57 F2 F6
  RHSUSES N41 N42 N43 N44 N45
ISAV
  LHSUSES A1 A5 A15 N21 F41
  NESTEDL A25 N29
  RHSUSES A15 A17 A19
ISAVW
  LHSUSES A14 A15 A17 A19 A25
  RHSUSES T7 T10 T13 T16 T21 T24 T27 -A14 -A15 -A17 -A19
ISCOMPREL
  LHSUSES -F31 F32C -B10 B10C
  RHSUSES T81 T82 T83 T86 T87 T88
ISCOP
  LHSUSES G10 G17 G18 G31 G32 A17 R1 N8C N15 N16
  NESTEDL A25 R5
  RHSUSES T1 T4
ISDEF
  LHSUSES A1 N9A N33
  RHSUSES N1 N2
ISIMPER
  LHSUSES N9E
  RHSUSES T71 T72 G41 G42 G43 G44 G45
ISINDEF
  LHSUSES A5 N31
  RHSUSES N5 N6 -N31
ISINDREL
  LHSUSES -F31 F32 -B10 B10I
  RHSUSES T31 T34
ISNOUN
  LHSUSES A16 R2 P1 N31 N33
  NESTEDL A25 R5 P9
  RHSUSES T86 G13 N21 N22 N23
ISNOUNW
  LHSUSES G13 N21 N22 N23 N29
  RHSUSES T41 T44 T47 T50 T53 T57 -G13 -N21 -N22 -N23
ISPRED
  LHSUSES -A15 R3 -R5 P2 -P9 F61
  NESTEDL -A25
  RHSUSES T81 T82 T83 A17
ISREL
  LHSUSES R11 R12 N9B
  RHSUSES R1 R2 R3
ISRELPRON
  LHSUSES P9 -N15 -N16
  RHSUSES P1 P2
ISRELPRONW
  LHSUSES P1 P2
  RHSUSES T60 T63 -P1 -P2
ISRELW
  LHSUSES R1 R2 R3 R5
  RHSUSES T31 T34 T37 T39 T81 T82 T83 T86 T87 T88 -R1 -R2 -R3
LEFTOF
  LHSUSES S0 S1 S4 S7 T1 T2 T4 T57 T81 T82 T83 E4 E6 G5 G10 G17 G18 A14 A15 A17
   A19 A25 R1 R2 R3 R5 P1 P2 P9 N9A N9B N9C N9D N9E N15 N16 N21 N22 N23 N29
  NESTEDL A25 M65 M66
  RHSUSES V1 T4 -T4
LOCAT
  LHSUSES Q1 -Q1 Q2 -Q2 Q2L Q3 Q6 Q7 Q35 Q45 Q57 Q61 Q64 Q65 Q66 Q70 Q71 F61
  F62 F63 F64 F65 F66 M84 -V53D V53L -V53R
  NESTEDL Q2 Q62 Q65 Q66 Q68 Q70 Q71
  RHSUSES Q1 -Q1 Q2 -Q2
LOCATERESULT
  LHSUSES Q76 Q77 Q78
  RHSUSES Q57 Q73 -Q76 -Q77 -Q78
LOCATESPACE
```

```
  LHSUSES Q57 Q57F Q81
  RHSUSES Q51 Q52 Q53 Q54 -Q57 -Q57F -Q81 W51
LOWX
  LHSUSES Q65 Q68 Q69
  RHSUSES Q62 Q65 -Q65 -Q68 -Q69
LOWY
  LHSUSES Q66 Q68 Q69
  RHSUSES Q62 Q66 -Q66 -Q68 -Q69
MAKESPACE
  LHSUSES Q81 Q82
  RHSUSES -Q81 -Q82 Q83 W27M
MAKESPACE2
  LHSUSES Q83
  RHSUSES -Q83
MAKESPACE3
  LHSUSES Q84 Q85
  RHSUSES Q83 -Q84 -Q85
MAKISA
  LHSUSES N41 N42 N43 N44 N45
  RHSUSES N31 -N41 -N42 -N43 -N44 -N45
MOVEHAND
  LHSUSES Q1 Q2 Q2L Q3
  RHSUSES -Q1 -Q2 -Q2L -Q3 Q32 Q35 Q46
NEWAV
  LHSUSES N51
  RHSUSES A5 -N51
NEWLOCAT
  LHSUSES Q6
  RHSUSES Q2 -Q6 -Q7
NEWLOCAT2
  LHSUSES Q7
  RHSUSES Q2 -Q7
NEWOBJ
  LHSUSES -F5 B11 -B18 B21 -B28
  RHSUSES N41 N42 N43 N44 N45
NEXT
  LHSUSES W0
  NESTEDL W0S W0T
  RHSUSES Q31 Q43 Q45 Q81 Q82 W1 W3 W4 W22 W23F W24 W27M W38 W38I W42 W45 W53
  W53A W54 W54A W56 W57
NEXTF
  LHSUSES W0F
  NESTEDL W0G
  RHSUSES W22 W23 W42 W48 W56 W57
NOCLEAR
  LHSUSES Q53
  NESTEDL Q51
  RHSUSES W22 -W22S
NPBOUND
  LHSUSES B51 B53 B55 B56 B57 B58
  RHSUSES S4 G45 N15 N16 B58 -B59 -B59I
NPBOUNDL
  LHSUSES B59 B59I
  RHSUSES G45 N15 N16 -B59 -B59I
NPGCHK
  LHSUSES N9
  RHSUSES T86 N1 N2 N5 N6
NPGCHK1
  LHSUSES N9A N9B N9C N9D N9E N10U
  RHSUSES N9 -N9A -N9B -N9C -N9D -N9E -N10U
NPGCHK2
  LHSUSES N10
  RHSUSES N9 -N10
NPGCHK3
  LHSUSES N10U
  RHSUSES N10 -N10U
NREPLY
  LHSUSES V0 V15
  RHSUSES S0 V0 -V0 V15 -V15
NRESTR
  LHSUSES F21 F23
  RHSUSES N33 -F21 -F23
NULLREF
  LHSUSES F51 F53 V12 V25
  RHSUSES F11 F23 F29
OCHK
  LHSUSES F11 F13 F15
  RHSUSES -F11 -F13 -F15 F21 F27 F31 F32 F32C F35 B18I
OLDAV
  LHSUSES -A1 -A5 -F41
  RHSUSES T81 T82 T83 A1 A5 F41
OLDREF
  LHSUSES -B1 -B3
  RHSUSES B1 B3
OLDREL
  LHSUSES -R11 -R12
  RHSUSES T81 T82 T83 R11 R12
PACK
  LHSUSES W51 W51S W54X
  RHSUSES W35 -W51 -W51S W53D -W54X W57F W58
PACKCHOICE
  LHSUSES W35 W53A W53D W54A W54X W54Z
  NESTEDL W51 W53A W53D W54A
  RHSUSES W53 W53A W53D W54 W54A -W54X -W54Z
PACKPUT
  LHSUSES W53 W53A W53D W54 W54A W58
  RHSUSES W51 -W53 -W53A -W53D -W54 -W54A -W58
PACKUPON
  LHSUSES W56 W57 W57F
  RHSUSES -W56 -W57 -W57F
PICKUP
  LHSUSES W1
  RHSUSES -W1 M81
PICKUP2
  LHSUSES W2
  RHSUSES -W2
PREDINCON
  LHSUSES B48 M2 M15 M53
  RHSUSES E21 -B48 -M2 -M15
PREDINCONT
  LHSUSES E21
  RHSUSES B28 B29
PREDREDUN
  LHSUSES B41 M5 M18
  RHSUSES E22 -B43 -B44 -M5 -M18
PREDREDUNT
  LHSUSES E22
  RHSUSES B25
PREDRESTR
  LHSUSES F35
  RHSUSES E23
PREDRESTRT
  LHSUSES E23
  RHSUSES B23 B24 B43 B44
PREDRESTRCHK
  LHSUSES B21 B23 B24 B25 B27 B28 B29
  RHSUSES F41 -B21 -B23 -B24 -B25 -B27 -B28 -B29 B48
PUT
  LHSUSES Q31
  RHSUSES -Q31 W12 W14 W25 W26 W53 W53A W54 W54A
PUTDOWN
  LHSUSES W10
  RHSUSES -W10 M84
PUTMOVE
  LHSUSES Q32
  RHSUSES -Q32
PUTON
  LHSUSES W20 W21 W23
  NESTEDL W20 W23
  RHSUSES -W21 -W23 M82 M83
PUTON1
  LHSUSES W24 W24F W26X
  RHSUSES W22 W23 -W24 -W24F W26D -W26X W34 W42 W48 W56 W57
PUTON1CHOICE
  LHSUSES W26 W26D W26X W26Z W34
  NESTEDL W24 W26 W26D
  RHSUSES W25 W26 W26D -W26X -W26Z
PUTONPUT
  LHSUSES W25 W26 W26D W27F
  RHSUSES W24 -W25 -W26 -W26D -W27F
PUTONSET
  LHSUSES W22 W22S
  RHSUSES W21 -W22 -W22S
PUTONSETO
  LHSUSES W21
  RHSUSES W20 -W21
PUTONSETCHOICE
  LHSUSES W36
  RHSUSES W20 -W36
QNOUN
  LHSUSES G13
  RHSUSES T87 -G13
QWFIND
  LHSUSES F1 F2
  RHSUSES G13 -F1 -F2
QWRDESCR2
```

```
  LHSUSES V18
  RHSUSES V17 -V18 V19
QWREPLY
  LHSUSES V15
  NESTEDL V15
  RHSUSES V10 V14 -V15
QWRPHRASE1
  LHSUSES D22 D23 D24
  NESTEDL D21
  RHSUSES D21 D22 -D22 D23 -D23 -D24
QWRPHRASE2
  LHSUSES D26 D27 D28
  NESTEDL D25
  RHSUSES D25 D26 -D26 D27 -D27 -D28
QWRREPLY1
  LHSUSES D21 D22 D23
  NESTEDL D21 D22 D23 D24
  RHSUSES V17 -D22 -D23
QWRREPLY2
  LHSUSES D25 D26 D27
  NESTEDL D26 D27 D28
  RHSUSES V18 -D26 -D27
QWRREPLY3
  LHSUSES D29
  RHSUSES V19 -D29
RAISEHAND
  LHSUSES Q35
  RHSUSES -Q35 W2
REFERS
  LHSUSES F23 F29 B1 B3 B15 B151 B18 B19 B25 B28 B29 B33 B331 B34 B341 B51 B53
  B591 M1 M2 M5 M63 M71 V10 V17 V19 V30 V31 V32 V35 V36 V37
  NESTEDL M87
  RHSUSES T88 N41 N42 N43 N44 N45 F13 -F23 -F29 B58C M83 -M83
RELINCON
  LHSUSES B38 B39 M1 M11 M51 M61 M62
  RHSUSES E31 -B38 -B39 -M1 -M11 -M61
RELINCONT
  LHSUSES E31
  RHSUSES B18 B19
RELREDUN
  LHSUSES B31 B311 M12 M64 M64A
  NESTEDL M64 M65
  RHSUSES E32 -B33 -B331 -B34 -B341 -M12
RELREDUNT
  LHSUSES E32
  RHSUSES B15 B151
RELRESTR
  LHSUSES F33
  RHSUSES E33
RELRESTR1
  LHSUSES F34 F341
  RHSUSES F33 -F34 -F341
RELRESTR2
  LHSUSES F31 F32 F32C
  RHSUSES F33
RELRESTRT
  LHSUSES E33
  RHSUSES B13 B131 B14 B33 B331 B34 B341
RELRESTRCHK
  LHSUSES B10 B10C B101
  RHSUSES B1 B3 -B10 -B10C -B101 B38 B39
RELRESTRCHK2
  LHSUSES B11 B13 B131 B14 B15 B151 B17 B18 B19 B19C B191
  RHSUSES B10 B10C B101 -B11 -B13 -B131 -B14 -B15 -B151 -B17 -B18 -B19 B19C
  B191
REMOHASREL
  LHSUSES Q11 Q23
  RHSUSES Q6 -Q23
REMOINSTACK
  LHSUSES Q13
  RHSUSES Q11 -Q13
REPLY
  LHSUSES V5
  RHSUSES V0 V15
REPLY0
  LHSUSES V0
  RHSUSES E2 E8 M81F M82P M83P M84F M85 M86F M87 -V0 V2 V12 V20 V25 V30 V31 V32
   V35 V36 V37 V51A V51H V51O V52A V52F V53D V53L V53R V54A V54F D24 D28 D29
RETRY
  LHSUSES -W11 -W12 W13 W14 W14D W15 W16 -W17 W18 W19 -W25 W26 W26D W26X -W53
  W53A W53D -W54 W54A W54X
  RHSUSES -W16 W17 -W26X W33 W34 W39 -W54X
SCAN
  LHSUSES -S1 -S4 T1 T2 T4 T7 T10 T13 T16 T21 T24 T27 T31 T34 T37 T39 T41 T44
  T47 T50 T53 T57 T60 T63 T66 T67 T71 T72 T81 T82 T83 T86 T87 T88 G1 G2 G5 G51
  G6 G7 G9 G10 G21 G41 G42 G43 G44 G45
  NESTEDL -S7
  RHSUSES S0 S1 -S7 -T1 -T2 -T4 -T7 -T10 -T13 -T16 -T21 -T24 -T27 -T31 -T34
  -T37 -T39 -T41 -T44 -T47 -T50 -T53 -T57 -T60 -T63 -T66 -T67 -T71 -T72 -T81
  -T82 -T83 -T86 -T87 -T88 -G1 -G2 -G5 -G51 -G6 -G7 -G9 -G10 -G21 -G41 -G42
  -G43 -G44 -G45
SCANFIN
  LHSUSES S0 S1 S4 S7 V8
  RHSUSES V1 S0 -S0 S1 -S1 -S4 -S7 -T67 T81 -T81 T82 -T82 T83 -T83 -V8
SENTBOUND
  LHSUSES M63 M64 M64A M65 M66 M71 M81 M81F M82 M82P M83 M83P M84 M84F M85 M86
  M86F M87 M89 V2 V10 V12 V14 V17 V19 V20 V30 V31 V32 V35 V36 V37 V40 V42 V44
  V46 V48 V49
  RHSUSES S4 M63 M81F M86F -M89
SENTENCE
  LHSUSES S4 G1 G2 G5 G51 G6 G7 G9 G13 B17 B18 B21 G41 B42 B43 B44 N15 N16 F53
  B17 B27 M11 M12 M51 M53
  RHSUSES Y1
STACKSET
  LHSUSES W41
  RHSUSES W40 -W41
STACKUP
  LHSUSES W40 W41
  NESTEDL W40
  RHSUSES -W40 -W41 M88
STACKUPSET
  LHSUSES W42 W43 W45 W46 W47 W48
  RHSUSES W41 -W42 W43 -W43 W44 -W45 W46 -W47 -W48
SUCCEED
  LHSUSES W0 W0S W0T
  RHSUSES Q32 Q41 Q47 Q84 -W0 W0S -W0S -W0T W2 W8 W22S W47 W48 W51S
TEST1
  LHSUSES Y1
TEST2
TEST3
TEXT
  LHSUSES S0
  RHSUSES Y1
TRACEPUTIN
  LHSUSES M83T
  RHSUSES M83 -M83T
TRACING
  RHSUSES Q1 Q2 Q2L Q11 Q13 Q15 Q17 Q31 Q43 Q45 Q47 Q47U Q49 Q57 Q57F Q62 Q63
  Q64 Q64B Q65 Q73 Q81 Q82 W0 W0F W0G W0S W0T W1 W3 W4 W10 W12 W14 W14D W15 W16
   W22 W23 W23F W24 W24F W25 W26 W26D W26X W27F W27M W33 W34 W35 W36 W38 W381
  W40 W42 W43 W44 W45 W46 W48 W53 W53A W53D W54 W54F W54X W56 W57 S0 T86 E11
  E12 E13 E21 E22 E23 E31 E32 E33 F13 F15 B58C M83 M83T
TRICDPACK
  LHSUSES -W51 -W56 -W57 W58
  NESTEDL -W51 -W51S -W56 -W57 -W57F
  RHSUSES W51 -W52B W56 W57 -W58
TRICDPUT
  LHSUSES -W22
  NESTEDL -W22 -W22S
  RHSUSES W22 -W22B
TRICDSTACK
  LHSUSES W42 -W42 W43 -W43 W45 -W45 W46 -W46 W48 -W48
  NESTEDL W42 -W42 -W43 W45 -W45 -W46 -W47 -W48
  RHSUSES W40 W42 -W42B W43 W45 W46
UNEVENT
  LHSUSES W31 W32
  NESTEDL W30
  RHSUSES Q1 Q2 Q47 Q47U Q49 W22 W22B -W31 -W32 W42 W42B W43 W45 W46 W51 W52B
  W56 W57 W58
UNGRASP
  LHSUSES Q49
  RHSUSES Q37 -Q49
USERESULT
  LHSUSES Q57F Q63 Q64B Q76 Q77 Q78
  RHSUSES Q51 Q52 Q53 Q54 -Q57F -Q63 -Q64B -Q76 -Q77 -Q78 W51
WBPINIT
  LHSUSES M89
  RHSUSES M81 M82 M83 M84 M86 -M89
WORDEQ
  LHSUSES E4 E8 G5 M8A
  RHSUSES T1 T4 T7 T10 T13 T16 T21 T24 T27 T31 T34 T37 T39 T41 T44 T47 T50 T53
  T57 T60 T63 T66 T71 T72 T81 T82 T83 T86 T87 T88 G1 G2 G5 G51 G6 G7 G9 G10 G21
   G41 G42 G43 G44 G45
```

```
FIRST SEGMENT

1 INPUT TEXT IS " PUT THE SMALL RED BLOCK ON THE BLUE BLOCK "
OBJ-1 AMBIG S3-1 BLOCK-1 PYRAMID-1 ...
OBJ-1 AMBIG R4-1 BLOCK-1 PYRAMID-3 ...
OBJ-1 REFERS BLOCK-1
OBJ-2 AMBIG B8-1 BLOCK-5 PYRAMID-2 ...
OBJ-2 REFERS BLOCK-5
RELINCON OBJ-1 BS-1 ON BLOCK-5 POS
STARTING GT PUTON BLOCK-1 ONTO BLOCK-5
GOAL G-1 CLEAROFF BLOCK-1
. GOAL G-2 GETRIDOF PYRAMID-1
  REJECTING (700 721 0)
  LOOKING AT (700 600 0)
  REGION AT (600 600 0) TOO SMALL
  LOOKING AT (706 9 0)
  FOUND REGION (600 0 0) TO (1200 600 0)
. . GOAL G-3 PUT PYRAMID-1 (900 451 0)
. . . GOAL G-4 GRASP PYRAMID-1
. . . . GOAL G-5 CLEAROFF PYRAMID-1
          G-5 SUCCEEDS
        (0) MOVING HAND FROM (0 100 400) TO (150 150 200)
        (1) GRASPING PYRAMID-1
        G-4 SUCCEEDS
      (2) LIFTING PYRAMID-1 FROM (100 100 100) TO (900 451 0)
      TAKING PYRAMID-1 FROM STACK-3
      STACK-3 DISMANTLED
      (3) LETTING GO OF PYRAMID-1
      ADDING PYRAMID-1 ON TABLE-1 (POS)
      G-3 SUCCEEDS
    G-2 SUCCEEDS
  G-1 SUCCEEDS
FOUND REGION CLEARTOP BLOCK-5
GOAL G-6 PUT BLOCK-1 (400 640 400)
. GOAL G-7 GRASP BLOCK-1
. . GOAL G-8 CLEAROFF BLOCK-1
      G-8 SUCCEEDS
    (4) MOVING HAND FROM (950 501 100) TO (150 150 100)
    (5) GRASPING BLOCK-1
    G-7 SUCCEEDS
  (6) LIFTING BLOCK-1 FROM (100 100 0) TO (400 640 400)
  (7) LETTING GO OF BLOCK-1
  ADDING BLOCK-1 ON BLOCK-5 (POS)
  MAKING STACK STACK-4 BLOCK-1 BLOCK-5
  G-6 SUCCEEDS
GT SUCCEEDS
2

REPLY (1 (OKAY))



CLEARTOP (BLOCK-1) (BLOCK-4) (PYRAMID-1) (PYRAMID-2) (PYRAMID-3)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
  (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
  (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
  (BLOCK-5 SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
  (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
  (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
  (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-1 ON BLOCK-5 POS) (BLOCK-2 ON TABLE-1 POS)
  (BLOCK-3 ON TABLE-1 POS) (BLOCK-4 ON BLOCK-3 POS) (BLOCK-5 ON TABLE-1 POS)
  (BOX-1 ON TABLE-1 POS) (PYRAMID-1 ON TABLE-1 POS) (PYRAMID-2 IN BOX-1 POS)
  (PYRAMID-3 ON BLOCK-2 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
  (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BOX-1 600 600 1)
  (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 240)
  (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-4) (BLOCK-2 STACK-2) (BLOCK-3 STACK-1) (BLOCK-4 STACK-1)
  (BLOCK-5 STACK-4) (PYRAMID-3 STACK-2)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
  (BLOCK-5 BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
  (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-1 400 640 400) (BLOCK-2 400 0 0) (BLOCK-3 0 300 0)
  (BLOCK-4 0 240 300) (BLOCK-5 300 640 0) (BOX-1 600 600 0) (HAND-1 450 890 500)
  (PYRAMID-1 900 451 0) (PYRAMID-2 640 640 1) (PYRAMID-3 500 100 200)
  (TABLE-1 0 0 0)

.........................................................................
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                 ++HI                                                  .
.          LBB5  SRB1           ++XILBP2                                .
.                                                                       .
.                                                                       .
.                                                          SGP1         .
.                                                                       .
LRB3                                                                    .
LGB4                                                                    .
.                                                                       .
.                                                                       .
.                      SRP3                                             .
.                 LGB2                                                  .
.........................................................................



PSBREAK :DEBUG AT :CYCLECMDS
NIL

(OK)



TRACE
(Y1-1 S0-1 G44-1
S1-1 G1-1 N2-1 N9-1 N9E-1 N10-1 FS-1 FS-2 FS-3 FS-4 FS-5 FS-6 FS-7 FS-8 FS-9
  FS-10 FS-11
S1-2 T27-1 A19-1 A1-1 F27-1 F27-2 F27-3 F27-4 F27-5 F27-6 F27-7 F27-8 F15-1
S1-3 T7-1 A15-1 A1-2 F27-9 F15-2
S1-4 T44-1 N21-1 N33-1 F21-1 F13-1
S1-5 T34-1 R2-1 R12-1
S1-6 G1-2 N1-1 N9-2 N98-1 N10-2 FS-12 FS-13 FS-14 FS-15 FS-16 FS-17 FS-18 FS-19
  FS-20 FS-21 FS-22
S1-7 T13-1 A19-2 A1-3 F27-10 F27-11 F27-12 F27-13 F27-14 F27-15 F27-16 F27-17
  F27-18 F15-3
S1-8 T44-2 N21-2 N33-2 F21-2 F13-2 B1-1 B10I-1 B18-1 E31-1 M61-1
S4-1 B55-1 B51-1 M71-1 M82-1 M89-1
W23-1 W24-1 W3-1 W11-1 O54-1 O61-1 O64-1 O64A-1 O64E-1 O64E-2 O64E-3 O62-1 O65-1
  O65-2 O66-1 O67-1 O69-1 O62-2 O65-3 O65-4 O67-2 O68-1 O71-1 O72-1 O73-1 O78-1
W12-1 O31-1 O45-1
W6-1 W8-1 O46-1 O1-1 O47-1
W8-2 O32-1 O2-1 O6-1 O23-1 O11-1 O13-1 O29-1 O7-1 O49-1 E12-1
W05-1 W8-3 W6-2 W8-4 O51-1 O57-1 O76-1
W25-1 O31-2 O45-2
W6-3 W8-5 O46-2 O1-2 O47-2
W8-6 O32-2 O2-2 O6-2 O29-2 O7-2 O49-2 E12-2 O27-1 O17-1 B10K-1
W05-2 W0T-1 V52-1 V52A-1 V8-1 B53-1)



2 INPUT TEXT IS " WHAT IS BELOW THE SMALL RED BLOCK "
OBJ-2 AMBIG S5-1 BLOCK-1 PYRAMID-1 ...
OBJ-2 AMBIG R6-1 BLOCK-1 PYRAMID-3 ...
OBJ-2 REFERS BLOCK-1
RELRESTP OBJ-1 W1-1 BELOW BLOCK-1 POS
OBJ-1 AMBIG W1-1 BLOCK-2 BLOCK-3 ...
2

REPLY (1 (THE BOX)) (2 (THE TABLE)) (3 (THE LARGE GREEN BLOCK))
  (4 (THE LARGE RED BLOCK)) (5 (THE LARGE GREEN BLOCK))
  (6 (THE LARGE BLUE BLOCK)) (7 (THE SMALL GREEN PYRAMID))
  (8 (THE LARGE BLUE PYRAMID)) (9 (THE SMALL RED PYRAMID))



CLEARTOP (BLOCK-1) (BLOCK-4) (PYRAMID-1) (PYRAMID-2) (PYRAMID-3)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
  (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
  (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
  (BLOCK-5 SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
  (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
  (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
  (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-1 ON BLOCK-5 POS) (BLOCK-2 ON TABLE-1 POS)
  (BLOCK-3 ON TABLE-1 POS) (BLOCK-4 ON BLOCK-3 POS) (BLOCK-5 ON TABLE-1 POS)
  (BOX-1 ON TABLE-1 POS) (PYRAMID-1 ON TABLE-1 POS) (PYRAMID-2 IN BOX-1 POS)
  (PYRAMID-3 ON BLOCK-2 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
  (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BOX-1 600 600 1)
  (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 240)
  (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-4) (BLOCK-2 STACK-2) (BLOCK-3 STACK-1) (BLOCK-4 STACK-1)
  (BLOCK-5 STACK-4) (PYRAMID-3 STACK-2)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
  (BLOCK-5 BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
  (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-1 400 640 400) (BLOCK-2 400 0 0) (BLOCK-3 0 300 0)
  (BLOCK-4 0 240 300) (BLOCK-5 300 640 0) (BOX-1 600 600 0) (HAND-1 450 630 500)
  (PYRAMID-1 900 451 0) (PYRAMID-2 840 640 1) (PYRAMID-3 500 100 200)
  (TABLE-1 0 0 0)

........................................................................
.                                                                      .
.                                                                      .
.                                                                      .
.                                                                      .
.                                                                      .
.                                                                      .
.                                                                      .
.                                                                      .
.                            ++H1                                      .
.                  LBB5  SRB1        ++X1LBP2                          .
.                                                                      .
.                                                                      .
.                                                     SGP1             .
.                                                                      .
LRB3                                                                   .
LGB4                                                                   .
.                                                                      .
.                                                                      .
.                              SRP3                                    .
.                        LGB2                                          .
........................................................................


PSBREAK :DEBUG AT :CYCLECMDS
NIL

(OK)



TRACE
(YZ-1 S0-2 TS7-1 G13-1 F1-1 F1-2 F1-3 F1-4 F1-5 F1-6 F1-7 F1-8 F1-9 F1-10 F1-11
S1-9 T1-1 G32-1 N16-1 BS9-1
S1-10 T88-1 R1-1 P11-1
S1-11 G1-3 N1-2 N9-3 N58-2 N10-3 FS-23 FS-24 FS-25 FS-26 FS-27 FS-28 FS-29 FS-30
  FS-31 FS-32 FS-33
S1-12 T27-2 A19-3 A1-4 F27-19 F27-20 F27-21 F27-22 F27-23 F27-24 F27-25 F27-26
  F15-4
S1-13 T7-2 A15-2 A1-5 F27-27 F15-5
S1-14 T44-3 N21-3 N33-3 F21-3 F13-3 B1-2 B1CC-1 F66-1 F66-2 F66-3 F66-4 F66-5
  F66-6 F66-7 F66-8 F66-9 B131-1 B131-2 B131-3 B131-4 B131-5 B131-6 B131-7
  B131-8 B131-9 E33-1 F33-1 F32C-1 F32C-2 F15-6
S4-2 BS3-2 BS5-2 V14-1 V14-2 V14-3 V14-4 V14-5 V14-6 V14-7 V14-8 V14-9 D1-1 D1-2
  D1-3 D1-4 D1-5 D1-6 D1-7 D1-8 D1-9 D11-1 D11-2 D11-3 D11-4 D11-5 D11-6 D11-7
  D2-1 D2-2 D2-3 D2-4 D2-5 D2-6 D2-7 D2-8 D2-9 D3-1 D3-2 D3-3 D3-4 D3-5 D3-6
  D3-7 D3-8 D3-9 D2-10 D2-11 D4-1 D4-2 V15-1 V15-2 D11-8 D11-9 D11-10 D11-11
  D11-12 D11-13 D11-14 D2-12 D2-13 D2-14 D2-15 D2-16 D2-17 D2-18 D4-3 D4-4 D4-5
  D4-6 D4-7 D4-8 D4-9 V15-3 V15-4 V15-5 V15-6 V15-7 V15-8 V15-9 BS1-2)


3 INPUT TEXT IS " PUT THE GREEN BLOCK TO THE RIGHT OF THE LARGE RED BLOCK IN
  THE BOX "
OBJ-1 AMBIG G3-1 BLOCK-2 BLOCK-4 ...
OBJ-1 AMBIG B4-1 BLOCK-2 BLOCK-4 ...
OBJ-2 AMBIG L10-1 BLOCK-2 BLOCK-3 ...
OBJ-2 REFERS BLOCK-3
RELRESTR OBJ-1 B4-1 TORIGHTOF BLOCK-3 POS
OBJ-1 REFERS BLOCK-2
OBJ-3 REFERS BOX-1
RELINCON OBJ-2 B12-1 IN BOX-1 POS
RELINCON OBJ-1 B12-1 IN BOX-1 POS
PUTIN STARTS WITH PUTON
STARTING GT PUTON BLOCK-2 ONTO BOX-1
GOAL G-1 CLEAROFF BLOCK-2
```

```
. GOAL G-2 GETRIDOF PYRAMID-3
  REJECTING (549 194 0)
  LOOKING AT (549 208 0)
  FOUND REGION (200 200 0) TO (300 451 0)
. . GOAL G-3 PUT PYRAMID-3 (200 233 0)
. . . GOAL G-4 GRASP PYRAMID-3
. . . . GOAL G-5 CLEAROFF PYRAMID-3
          G-5 SUCCEEDS
        (0) MOVING HAND FROM (450 600 500) TO (500 100 440)
        (1) GRASPING PYRAMID-3
        G-4 SUCCEEDS
      (2) LIFTING PYRAMID-3 FROM (500 100 200) TO (200 233 0)
      TAKING PYRAMID-3 FROM STACK-2
      STACK-2 DISMANTLED
      (3) LETTING GO OF PYRAMID-3
      ADDING PYRAMID-3 ON TABLE-1 (POS)
      G-3 SUCCEEDS
    G-2 SUCCEEDS
  G-1 SUCCEEDS
LOOKING AT (951 1012 1)
FOUND REGION (940 840 1) TO (1200 1200 1)
GOAL G-6 PUT BLOCK-2 (940 840 1)
. GOAL G-7 GRASP BLOCK-2
. . GOAL G-8 CLEAROFF BLOCK-2
      G-8 SUCCEEDS
    (4) MOVING HAND FROM (250 283 240) TO (500 100 200)
    (5) GRASPING BLOCK-2
    G-7 SUCCEEDS
  (6) LIFTING BLOCK-2 FROM (400 0 0) TO (940 840 1)
  (7) LETTING GO OF BLOCK-2
  ADDING BLOCK-2 ON BOX-1 (POS)
  ADDING BLOCK-2 IN BOX-1 (POS)
  G-6 SUCCEEDS
GT SUCCEEDS
2

REPLY (1 (OKAY))


CLEARTOP (BLOCK-1) (BLOCK-2) (BLOCK-4) (PYRAMID-1) (PYRAMID-2) (PYRAMID-3)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
  (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
  (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
  (BLOCK-5 SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
  (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
  (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
  (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-1 ON BLOCK-5 POS) (BLOCK-2 IN BOX-1 POS) (BLOCK-3 ON TABLE-1 POS)
  (BLOCK-4 ON BLOCK-3 POS) (BLOCK-5 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS)
  (PYRAMID-1 ON TABLE-1 POS) (PYRAMID-2 IN BOX-1 POS) (PYRAMID-3 ON TABLE-1 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
  (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BOX-1 600 600 1)
  (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 240)
  (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-4) (BLOCK-3 STACK-1) (BLOCK-4 STACK-1) (BLOCK-5 STACK-4)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
  (BLOCK-5 BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
  (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-1 400 640 400) (BLOCK-2 940 840 1) (BLOCK-3 0 300 0)
  (BLOCK-4 0 240 300) (BLOCK-5 300 640 0) (BOX-1 600 600 0)
  (HAND-1 1040 940 201) (PYRAMID-1 900 451 0) (PYRAMID-2 640 640 1)
  (PYRAMID-3 200 233 0) (TABLE-1 0 0 0)

.......................................................................
.                                                                     .
.                                                                     .
.                                                                     .
.                                                           ++H1      .
.                                                     LGB2            .
.                                                                     .
.                                                                     .
.                                                                     .
.                LBB5  SRB1         ++X1LBP2                          .
.                                                                     .
.                                                                     .
.                                                   SGP1              .
.                                                                     .
LRB3                                                                  .
LGB4                                                                  .
.         SRP3                                                        .
.                                                                     .
```

```
.
.
..........................................................................

PSBREAK (DEBUG AT (CYCLECMDS
NIL

(COPE)
CORE (FREE.FULL): (4045 . 1000)
ok

(OK)



2

RUN TIME 7 MIN. 21.9 SEC

EXAM      TRY      FIRE     WMACT   E/F     E/T     T/F
3474      740      571      1772    6.08    4.69    1.30
0.127     0.597    0.774    0.249   SEC AVG

1000 INSERTS 604 DELETES 201 WARNINGS 14 NEW OBJECTS
MAX (SMPX LENGTH 205
CORE (FREE.FULL): (4509 . 1919) USED (3294 . 555)

(ACTS LOADPS (WBLOX . EXP) (MILIM . MAC) (WBLOOS . EXP) (WBLOW . EXP) (MILIPS
   . EXP) LOADPS (MILIPW . EXP) (MILGARP . EXP) (MILN . EXP) (MILFB . EXP) (
  MILM . EXP) (MILVO . EXP) MILC LOADPS (MILY8 . EXP) SAVEPS (CLOSED (WBLOX .
  EXP)) RUN SMPXEMPTY SAVEDB (CLOSED (WBL1 . DBS)) (CLOSED (WBL1 . TRS)) SAVEDB
  RUN SMPXEMPTY SMPXEMPTY SMPXEMPTY

TRACE
(Y3-1 S0-3 G44-2
S1-15 G1-4 N2-2 N9-4 N9E-2 N10-4 FS-34 FS-35 FS-36 FS-37 FS-38 FS-39 FS-40 FS-41
  FS-42 FS-43 FS-44
S1-16 T10-1 A19-4 A1-6 F27-28 F27-29 F27-30 F27-31 F27-32 F27-33 F27-34 F27-35
  F15-7
S1-17 T44-4 N21-4 N33-4 F21-4 F15-8
S1-18 T82-1 R2-2 P3-1 P12-2
S1-19 G1-5 N1-3 N9-5 N98-3 N10-5 FS-45 FS-46 FS-47 FS-48 FS-49 FS-50 FS-51 FS-52
  FS-53 FS-54 FS-55
S1-20 T21-1 A19-5 A1-7 F27-36 F27-37 F27-38 F27-39 F27-40 F27-41 F15-9
S1-21 T7-3 A15-3 A1-8 F27-42 F27-43 F27-44 F27-45 F13-4 B1-3 B10C-2 F82-1
  B13I-10 E33-2 F33-2 F32C-3 F13-5
S1-22 T44-5 N21-5 N33-5
S1-23 T31-1 R2-3 P12-3
S1-24 G1-6 N1-4 N9-6 N98-4 N10-6 FS-56 FS-57 FS-58 FS-59 FS-60 FS-61 FS-62 FS-63
  FS-64 FS-65 FS-66
S1-25 T53-1 N22-1 N33-6 F21-5 F21-6 F21-7 F21-8 F21-9 F21-10 F21-11 F21-12
  F21-13 F21-14 F13-6 B1-4 B10I-2 B10J-1 B10-2 E31-2 B30-1 B?3I-3 B10J-2 B10-3
  E31-3 M61-2
S4-3 B55-3 B51-3 M71-2 M83-1 M83T-1 M89-2
W23-2 W24-2 W3-2 W11-2 O54-2 O61-2 O64-2 O64A-2 O64E-4 O64E-5 O64E-6 O62-3 O65-5
  O66-2 O67-3 O68-2 O70-1 O71-2 O72-2 O73-2 O79-2
W12-2 O31-3 O45-3
W6-4 W8-7 O46-3 O1-3 O47-3
W8-8 O32-3 O2-3 O6-3 O11-2 O13-2 O23-2 O29-3 O7-3 O49-3 E12-3
W05-3 W8-9 W6-5 W8-10 O52-1 O61-3 O62-4 O65-6 O66-3 O67-4 O68-3 O72-3 O73-3
  O77-1
W25-2 O31-4 O45-4
W6-6 W8-11 O46-4 O1-4 O47-4
W8-12 O32-4 O2-4 O6-4 O29-4 O7-4 O49-4 E12-4 O21-1 E12-5
W05-4 W0T-2 V52-2 V52A-2 V8-2 B53-3)

FIRED 99 OUT OF 400 PRODS


                  - - - - - - - - - - - - - - - - - - - -


(FIRST SEGMENT - FIRST TEST)

          Y S T E N G A R F B M V O W

Y1-1      Y                             1.
S0-1        S                           1.
G44-1               G                   1.
S1-1        S                           1.
```

```
G1-1                G                   1.
N2-1              N                     4....
FS-1                      F             11...........
S1-2        S                           1.
T27-1         T                         1.
A19-1                 A                 2..
F27-1                     F             9.........
S1-3        S                           1.
T7-1          T                         1.
A15-1                 A                 2..
F27-9                     F             2..
S1-4        S                           1.
T44-1         T                         1.
N21-1             N                     2..
F21-1                     F             2..
S1-5        S                           1.
T34-1         T                         1.
R2-1                    R               2..
S1-6        S                           1.
G1-2                G                   1.
N1-1              N                     4....
FS-12                     F             11...........
S1-7        S                           1.
T13-1         T                         1.
A19-2                 A                 2..
F27-10                    F             10..........
S1-8        S                           1.
T44-2         T                         1.
N21-2             N                     2..
F21-2                     F             2..
B1-1                        B           3...
E31-1           E                       1.
M61-1                         M         1.
S4-1        S                           1.
B55-1                       B           2..
M71-1                         M         3...
W23-1                               W   4....
O54-1                             O     22......................
W12-1                               W   1.
O31-1                             O     2..
W6-1                                W   2..
O46-1                             O     3...
W8-2                                W   1.
O32-1                             O     9.........
E12-1           E                       1.
W05-1                               W   4....
O51-1                             O     3...
W25-1                               W   1.
O31-2                             O     2..
W6-3                                W   2..
O46-2                             O     3...
W8-6                                W   1.
O32-2                             O     6......
E12-2           E                       1.
O27-1                             O     2..
B1AK-1                      B           1.
W05-2                               W   2..
V52-1                           V       3...
B53-1                       B           1.


PERCENTAGES OF FIRINGS OF EACH TYPE, OUT OF TOTAL 174
Y 0
S 5.....
T 3...
E 1.
N 6......
G 1.
A 3...
R 1.
F 27...........................
B 4....
M 2..
V 1.
O 29.............................
W 10..........


              - - - - - - - - - - - - - - - - - - - -


SECOND SEGMENT
```

```
4 INPUT TEXT IS " PUT THE GREEN BLOCK ON THE BLOCK IN THE BOX "
OBJ-1 AMBIG G3-1 BLOCK-2 BLOCK-4 ...
OBJ-1 AMBIG B4-1 BLOCK-2 BLOCK-4 ...
OBJ-2 AMBIG B7-1 BLOCK-1 BLOCK-2 ...
OBJ-3 REFERS BOX-1
RELRESTR OBJ-2 B7-1 IN BOX-1 POS
OBJ-2 REFERS BLOCK-2
OBJ-1 REFERS BLOCK-4
RELINCON OBJ-1 B4-1 ON BLOCK-2 POS
STARTING GT PUTON BLOCK-4 ONTO BLOCK-2
GOAL G-1 CLEAROFF BLOCK-4
  G-1 SUCCEEDS
FOUND REGION CLEARTOP BLOCK-2
GOAL G-2 PUT BLOCK-4 (940 840 201)
. GOAL G-3 GRASP BLOCK-4
. . GOAL G-4 CLEAROFF BLOCK-4
      G-4 SUCCEEDS
    (0) MOVING HAND FROM (1040 940 201) TO (100 340 500)
    (1) GRASPING BLOCK-4
    G-3 SUCCEEDS
  (2) LIFTING BLOCK-4 FROM (0 240 300) TO (940 840 201)
  TAKING BLOCK-4 FROM STACK-1
  STACK-1 DISMANTLED
  (3) LETTING GO OF BLOCK-4
  MAKING STACK STACK-5 BLOCK-4 BLOCK-2
  ADDING BLOCK-4 ON BLOCK-2 (POS)
  G-2 SUCCEEDS
GT SUCCEEDS
2

REPLY (1 (OKAY))
```

```
.............................................................................
.                                                                           .
.                                                                           .
.                                                                           .
.                                                      LGB4  ++H1           .
.                                                      LGB2                 .
.                                                                           .
.                                                                           .
.                                                                           .
.                 LBB5  SRB1          ++XILBP2                              .
.                                                                           .
.                                                                           .
.                                                     SGP1                  .
.                                                                           .
LRB3                                                                        .
.                                                                           .
.         SRP3                                                              .
.                                                                           .
.                                                                           .
.                                                                           .
.............................................................................
```

```
5 INPUT TEXT IS " WHAT IS IN THE BOX "
OBJ-2 REFERS BOX-1
RELRESTR OBJ-1 W1-1 IN BOX-1 POS
OBJ-1 AMBIG W1-1 BLOCK-2 BLOCK-4 ...
2

REPLY (1 (THE LARGE GREEN BLOCK)) (2 (THE LARGE GREEN BLOCK))
  (3 (THE LARGE BLUE PYRAMID))
```

```
.............................................................................
.                                                                           .
.                                                                           .
.                                                                           .
.                                                      LGB4  ++H1           .
.                                                      LGB2                 .
.                                                                           .
```

```
.
.                 LBB5  SRB1          ++XILBP2
.
.
.                                                                SGP1
.
LRB3
.
.         SRP3
.
.
.
..........................................................................
```

```
6 INPUT TEXT IS " WHAT IS GREEN "
PREDRESTR OBJ-1 G3-1 COLOR GREEN POS
OBJ-1 AMBIG G3-1 BLOCK-2 BLOCK-4 ...
2

REPLY (1 (THE LARGE GREEN BLOCK)) (2 (THE LARGE GREEN BLOCK))
  (3 (THE SMALL GREEN PYRAMID))
```

```
..........................................................................
.
.
.
.
.                                                                LGB4  ++H1
.                                                                LGB2
.
.
.                 LBB5  SRB1          ++XILBP2
.
.
.                                                                SGP1
.
LRB3
.
.         SRP3
.
.
.
..........................................................................
```

```
7 INPUT TEXT IS " PUT THE GREEN PYRAMID AND THE RED PYRAMID ON THE BLUE BLOCK "
OBJ-1 AMBIG G3-1 BLOCK-2 BLOCK-4 ...
OBJ-1 REFERS PYRAMID-1
OBJ-2 AMBIG R7-1 BLOCK-1 BLOCK-3 ...
OBJ-2 REFERS PYRAMID-3
OBJ-3 AMBIG B11-1 BLOCK-5 PYRAMID-2 ...
OBJ-3 REFERS BLOCK-5
RELINCON OBJ-2 P8-1 ON BLOCK-5 POS
DOING GT PUTON SET S-2 (BLOCK-5)
GOAL G-1 PUTON PYRAMID-1 ONTO BLOCK-5
. GOAL G-2 CLEAROFF PYRAMID-1
    G-2 SUCCEEDS
  REJECTING (417 665 400)
  LOOKING AT (400 665 400)
  FOUND REGION (300 640 400) TO (400 740 400)
. GOAL G-3 PUT PYRAMID-1 (300 640 400)
. . GOAL G-4 GRASP PYRAMID-1
. . . GOAL G-5 CLEAROFF PYRAMID-1
        G-5 SUCCEEDS
      (1) MOVING HAND FROM (1040 940 401) TO (550 501 100)
      (2) GRASPING PYRAMID-1
      G-4 SUCCEEDS
    (3) LIFTING PYRAMID-1 FROM (500 451 0) TO (300 640 400)
    (4) LETTING GO OF PYRAMID-1
    ADDING PYRAMID-1 ON BLOCK-5 (POS)
    ADDING PYRAMID-1 TO STACK-4
    G-3 SUCCEEDS
  G-1 SUCCEEDS
DOING GT PUTON SET S-2 (BLOCK-5)
GOAL G-6 PUTON PYRAMID-3 ONTO BLOCK-5
. GOAL G-7 CLEAROFF PYRAMID-3
    G-7 SUCCEEDS
  REJECTING (456 659 400)
  LOOKING AT (500 659 400)
  FOUND REGION (500 640 400) TO (600 740 400)
. GOAL G-8 PUT PYRAMID-3 (500 640 400)
. . GOAL G-9 GRASP PYRAMID-3
. . . GOAL G-10 CLEAROFF PYRAMID-3
        G-10 SUCCEEDS
      (6) MOVING HAND FROM (350 690 500) TO (250 283 240)
      (7) GRASPING PYRAMID-3
      G-9 SUCCEEDS
    (8) LIFTING PYRAMID-3 FROM (200 233 0) TO (500 640 400)
    (9) LETTING GO OF PYRAMID-3
    ADDING PYRAMID-3 TO STACK-4
    ADDING PYRAMID-3 ON BLOCK-5 (POS)
    G-8 SUCCEEDS
  G-6 SUCCEEDS
GT SUCCEEDS
2

REPLY (1 (OKAY))
```

```
.........................................................................
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                           LGB4        .
.                                                           LGB2        .
.                                                                       .
.                                                                       .
.                   SGP1            ++H1                                .
.                   LBB6  SRB1  SRP3    ++X1LBP2                        .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
LRB3                                                                    .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.........................................................................
```

```
PSBREAK :DEBUG AT :CYCLECMDS
NIL

(CORE)
CORE (FREE.FULL): (4926 . 2152)
ok

(OK)



2

RUN TIME 5 MIN. 41.7 SEC

EXAM     TRY     FIRE     WMACT   E/F    E/T    T/F
3398     700     559      1713    6.08   4.85   1.25
0.101   0.400   0.611    0.199   SEC AVG

1050 INSERTS 663 DELETES 302 WARNINGS 14 NEW OBJECTS
MAX :SMPX LENGTH 225
CORE (FREE.FULL): (5835 . 2176) USED (2410 . 295)

:ACTS SAVEDB RUN SMPXEMPTY SMPXEMPTY SMPXEMPTY SMPXEMPTY
FIRED 82 OUT OF 409 PRODS
```

- -

```
THIRD SEGMENT

8 INPUT TEXT IS " WHAT IS ON THE TABLE "
OBJ-2 REFERS TABLE-1
RELRESTR OBJ-1 W1-1 ON TABLE-1 POS
OBJ-1 AMBIG W1-1 BLOCK-3 BLOCK-5 ...
2

REPLY (1 (THE BOX)) (2 (THE LARGE RED BLOCK)) (3 (THE LARGE BLUE BLOCK))
```

```
.........................................................................
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                           LGB4        .
.                                                           LGB2        .
.                                                                       .
.                                                                       .
.                   SGP1            ++H1                                .
.                   LBB5  SRB1  SRP3    ++X1LBP2                        .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
LRB3                                                                    .
.                                                                       .
.                                                                       .
.
.
.........................................................................
```

```
9 INPUT TEXT IS " PUT THE LARGE RED BLOCK AND THE GREEN PYRAMID IN THE BOX "
OBJ-1 AMBIG L3-1 BLOCK-2 BLOCK-3 ...
OBJ-1 REFERS BLOCK-3
OBJ-2 AMBIG G8-1 BLOCK-2 BLOCK-4 ...
OBJ-2 REFERS PYRAMID-1
OBJ-3 REFERS BOX-1
RELINCON OBJ-2 P9-1 IN BOX-1 POS
PUTIN STARTS WITH PUTON
DOING GT PUTON SET S-2 (BOX-1)
GOAL G-1 PUTON BLOCK-3 ONTO BOX-1
. GOAL G-2 CLEAROFF BLOCK-3
    G-2 SUCCEEDS
  REJECTING (693 698 1)
  LOOKING AT (640 698 1)
  REGION AT (600 600 1) TOO SMALL
  REJECTING (960 997 1)
  LOOKING AT (940 997 1)
  REGION AT (940 840 1) TOO SMALL
  LOOKING AT (726 961 1)
  FOUND REGION (600 840 1) TO (940 1200 1)
. GOAL G-3 PUT BLOCK-3 (600 840 1)
. . GOAL G-4 GRASP BLOCK-3
. . . GOAL G-5 CLEAROFF BLOCK-3
        G-5 SUCCEEDS
      (1) MOVING HAND FROM (550 690 640) TO (100 450 300)
      (2) GRASPING BLOCK-3
      G-4 SUCCEEDS
    (3) LIFTING BLOCK-3 FROM (0 300 0) TO (600 840 1)
    (4) LETTING GO OF BLOCK-3
    ADDING BLOCK-3 IN BOX-1 (POS)
    G-3 SUCCEEDS
  G-1 SUCCEEDS
DOING GT PUTON SET S-2 (BOX-1)
GOAL G-6 PUTON PYRAMID-1 ONTO BOX-1
. GOAL G-7 CLEAROFF PYRAMID-1
    G-7 SUCCEEDS
  REJECTING (971 935 1)
  LOOKING AT (940 935 1)
  REGION AT (940 840 1) TOO SMALL
  LOOKING AT (1121 717 1)
  FOUND REGION (940 600 1) TO (1200 840 1)
. GOAL G-8 PUT PYRAMID-1 (940 600 1)
. . GOAL G-9 GRASP PYRAMID-1
. . . GOAL G-10 CLEAROFF PYRAMID-1
        G-10 SUCCEEDS
      (6) MOVING HAND FROM (700 550 301) TO (350 630 500)
      (7) GRASPING PYRAMID-1
      G-9 SUCCEEDS
    (8) LIFTING PYRAMID-1 FROM (300 640 400) TO (940 800 1)
    TAKING PYRAMID-1 FROM STACK-4
    (9) LETTING GO OF PYRAMID-1
    ADDING PYRAMID-1 ON BOX-1 (POS)
    ADDING PYRAMID-1 IN BOX-1 (POS)
    G-8 SUCCEEDS
  G-6 SUCCEEDS
GT SUCCEEDS
2

REPLY (1 (OKAY))
```

```
                                                LGB4
                               LRB3             LGB2



          LBB5  SRB1  SRP3    **X1LBP2          SGP1**H1
```

```
10 INPUT TEXT IS " WHAT IS TO THE LEFT OF THE BOX "
OBJ-2 REFERS BOX-1
RELRESTR OBJ-1 WI-1 TOLEFTOF BOX-1 POS
OBJ-1 AMBIG WI-1 BLOCK-1 BLOCK-5 ...
2

REPLY (1 (THE SMALL RED BLOCK)) (2 (THE LARGE BLUE BLOCK))
  (3 (THE SMALL RED PYRAMID)) (4 (THE TABLE))
```

```
                                                LGB4
                               LRB3             LGB2



          LBB5  SRB1  SRP3    **X1LBP2          SGP1**H1
```

```
11 INPUT TEXT IS " WHAT IS IN FRONT OF THE BOX "
OBJ-2 REFERS BOX-1
RELRESTR OBJ-1 WI-1 INFRONTOF BOX-1 POS
```

```
OBJ-1 REFERS TABLE-1
2

REPLY (1 (THE TABLE))


CLEARTOP (BLOCK-1) (BLOCK-3) (BLOCK-4) (PYRAMID-1) (PYRAMID-2) (PYRAMID-3)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
  (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
  (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
  (BLOCK-5 SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
  (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
  (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
  (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-1 ON BLOCK-5 POS) (BLOCK-2 IN BOX-1 POS) (BLOCK-3 IN BOX-1 POS)
  (BLOCK-4 ON BLOCK-2 POS) (BLOCK-5 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS)
  (PYRAMID-1 IN BOX-1 POS) (PYRAMID-2 IN BOX-1 POS) (PYRAMID-3 ON BLOCK-5 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
  (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BOX-1 600 600 1)
  (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 200)
  (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-4) (BLOCK-2 STACK-5) (BLOCK-4 STACK-5) (BLOCK-5 STACK-4
  (PYRAMID-3 STACK-4)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
  (BLOCK-5 BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
  (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-1 400 640 400) (BLOCK-2 940 840 1) (BLOCK-3 600 840 1)
  (BLOCK-4 940 840 201) (BLOCK-5 300 640 0) (BOX-1 600 600 0)
  (HAND-1 950 650 101) (PYRAMID-1 940 600 1) (PYRAMID-2 640 640 1)
  (PYRAMID-3 500 640 400) (TABLE-1 0 0 0)
```

```
                                                LGB4
                               LRB3             LGB2



          LBB5  SRB1  SRP3    **X1LBP2          SGP1**H1
```

```
RUN TIME 6 MIN. 11.4 SEC

EXAM     TRY      FIRE     WMACT   E/F     E/T     T/F
3093     647      547      1700    5.65    4.78    1.18
0.120    0.574    0.679    0.210   SEC AVG

1010 INSCRTS 602 DELETES 314 WARNINGS 14 NEW OBJECTS
MAX :SMPX LENGTH 156
CORE (FREE.FULL): (6012 . 2321) USED (1393 . 144)
FIRED 42 OUT OF 400 PRODS


                    - - - - - - - - - - - - - - - - - - - -


FOURTH SEGMENT

12 INPUT TEXT IS " PUT A SMALL PYRAMID AND A SMALL PYRAMID AND A GREEN BLOCK
  AND THE SMALL RED BLOCK ON THE LARGE RED BLOCK "
OBJ-1 AMBIG S3-1 BLOCK-1 PYRAMID-1 ...
OBJ-1 AMBIG P4-1 PYRAMID-1 PYRAMID-3 ...
CHOOSING PYRAMID-3 FOR OBJ-1
OBJ-2 AMBIG S7-1 BLOCK-1 PYRAMID-1 ...
OBJ-2 AMBIG P8-1 PYRAMID-1 PYRAMID-3 ...
CHOOSING PYRAMID-1 FOR OBJ-2
OBJ-3 AMBIG G11-1 BLOCK-2 BLOCK-4 ...
```

```
OBJ-3 AMBIG B12-1 BLOCK-2 BLOCK-4 ...
CHOOSING BLOCK-4 FOR OBJ-3
OBJ-4 AMBIG S15-1 BLOCK-1 PYRAMID-1 ...
OBJ-4 AMBIG R16-1 BLOCK-1 PYRAMID-3 ...
OBJ-4 REFERS BLOCK-1
OBJ-5 AMBIG L20-1 BLOCK-2 BLOCK-3 ...
OBJ-5 REFERS BLOCK-3
RELINCON OBJ-4 B17-1 ON BLOCK-3 POS
DOING GT PUTON SET S-2 (BLOCK-3)
GOAL G-1 PUTON BLOCK-4 ONTO BLOCK-3
. GOAL G-2 CLEAROFF BLOCK-4
    G-2 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-3
. GOAL G-3 PUT BLOCK-4 (600 840 301)
. . GOAL G-4 GRASP BLOCK-4
. . . GOAL G-5 CLEAROFF BLOCK-4
        G-5 SUCCEEDS
      (1) MOVING HAND FROM (990 650 101) TO (1040 940 401)
      (2) GRASPING BLOCK-4
      G-4 SUCCEEDS
    (3) LIFTING BLOCK-4 FROM (940 840 201) TO (600 840 301)
    TAKING BLOCK-4 FROM STACK-5
    STACK-5 DISMANTLED
    (4) LETTING GO OF BLOCK-4
    MAKING STACK STACK-6 BLOCK-4 BLOCK-3
    ADDING BLOCK-4 ON BLOCK-3 (POS)
    G-3 SUCCEEDS
  G-1 SUCCEEDS
DOING GT PUTON SET S-2 (BLOCK-3)
GOAL G-6 PUTON BLOCK-1 ONTO BLOCK-3
. GOAL G-7 CLEAROFF BLOCK-1
    G-7 SUCCEEDS
  REJECTING (654 982 301)
  LOOKING AT (654 1040 301)
  FOUND REGION (600 1040 301) TO (800 1140 301)
. GOAL G-8 PUT BLOCK-1 (600 1040 301)
. . GOAL G-9 GRASP BLOCK-1
. . . GOAL G-10 CLEAROFF BLOCK-1
        G-10 SUCCEEDS
      (6) MOVING HAND FROM (700 940 501) TO (450 690 500)
      (7) GRASPING BLOCK-1
      G-9 SUCCEEDS
    (8) LIFTING BLOCK-1 FROM (400 640 400) TO (600 1040 301)
    TAKING BLOCK-1 FROM STACK-4
    (9) LETTING GO OF BLOCK-1
    ADDING BLOCK-1 ON BLOCK-3 (POS)
    ADDING BLOCK-1 TO STACK-6
    G-8 SUCCEEDS
  G-6 SUCCEEDS
DOING GT PUTON SET S-2 (BLOCK-3)
GOAL G-11 PUTON PYRAMID-1 ONTO BLOCK-3
. GOAL G-12 CLEAROFF PYRAMID-1
    G-12 SUCCEEDS
  REJECTING (640 1006 301)
  LOOKING AT (640 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (664 885 301)
  LOOKING AT (664 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (630 991 301)
  LOOKING AT (630 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  LOOKING AT (731 1052 301)
  FOUND REGION (700 1040 301) TO (800 1140 301)
. GOAL G-13 PUT PYRAMID-1 (700 1040 301)
. . GOAL G-14 GRASP PYRAMID-1
. . . GOAL G-15 CLEAROFF PYRAMID-1
        G-15 SUCCEEDS
      (11) MOVING HAND FROM (650 1090 401) TO (990 650 101)
      (12) GRASPING PYRAMID-1
      G-14 SUCCEEDS
    (13) LIFTING PYRAMID-1 FROM (940 600 1) TO (700 1040 301)
    (14) LETTING GO OF PYRAMID-1
    ADDING PYRAMID-1 TO STACK-6
    ADDING PYRAMID-1 ON BLOCK-3 (POS)
    G-13 SUCCEEDS
  G-11 SUCCEEDS
DOING GT PUTON SET S-2 (BLOCK-3)
GOAL G-16 PUTON PYRAMID-3 ONTO BLOCK-3
. GOAL G-17 CLEAROFF PYRAMID-3
    G-17 SUCCEEDS
  REJECTING (615 1020 301)
  LOOKING AT (615 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (676 914 301)
  LOOKING AT (676 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (600 910 301)
  LOOKING AT (600 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (646 1046 301)
  LOOKING AT (646 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (617 1074 301)
  LOOKING AT (617 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  FINDSPACE LIMIT EXCEEDED
  NO SPACE TO PUTON PYRAMID-3 BLOCK-3
  G-16 FAILS
(14) GRASPING PYRAMID-1
(13) LIFTING PYRAMID-1 FROM (700 1040 301) TO (940 600 1)
TAKING PYRAMID-1 FROM STACK-6
ADDING PYRAMID-1 ON BOX-1 (POS)
ADDING PYRAMID-1 IN BOX-1 (POS)
(12) LETTING GO OF PYRAMID-1
(11) MOVING HAND FROM (990 650 101) TO (650 1090 401)
GOAL G-11 RETRY PUTON1 PYRAMID-1 BLOCK-3
. GOAL G-18 CLEAROFF PYRAMID-1
    G-18 SUCCEEDS
  REJECTING (714 931 301)
  LOOKING AT (714 1040 301)
  FOUND REGION (700 1040 301) TO (800 1140 301)
  FOUNDSPACE DUPLICATED (700 1040 301)
. GOAL G-19 CLEAROFF PYRAMID-1
    G-19 SUCCEEDS
  REJECTING (666 904 301)
  LOOKING AT (666 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (661 1025 301)
  LOOKING AT (661 1040 301)
  REGION AT (600 1040 301) TOO SMALL
  REJECTING (703 932 301)
  LOOKING AT (703 1040 301)
  FOUND REGION (700 1040 301) TO (800 1140 301)
  FOUNDSPACE DUPLICATED (700 1040 301)
  G-11 EXHAUSTED
  (9) GRASPING BLOCK-1
  (8) LIFTING BLOCK-1 FROM (600 1040 301) TO (400 640 400)
  TAKING BLOCK-1 FROM STACK-6
  ADDING BLOCK-1 ON BLOCK-5 (POS)
  ADDING BLOCK-1 TO STACK-4
  (7) LETTING GO OF BLOCK-1
  (6) MOVING HAND FROM (450 690 500) TO (700 940 501)
GOAL G-6 RETRY PUTON1 BLOCK-1 BLOCK-3
. GOAL G-20 CLEAROFF BLOCK-1
    G-20 SUCCEEDS
  REJECTING (675 950 301)
  LOOKING AT (675 1040 301)
  FOUND REGION (600 1040 301) TO (800 1140 305)
  FOUNDSPACE DUPLICATED (600 1040 301)
. GOAL G-21 CLEAROFF BLOCK-1
    G-21 SUCCEEDS
  REJECTING (684 1004 301)
  LOOKING AT (684 1040 301)
  FOUND REGION (600 1040 301) TO (800 1140 301)
  FOUNDSPACE DUPLICATED (600 1040 301)
  G-6 EXHAUSTED
  (4) GRASPING BLOCK-4
  (3) LIFTING BLOCK-4 FROM (600 840 301) TO (940 840 201)
  TAKING BLOCK-4 FROM STACK-6
  STACK-6 DISMANTLED
  ADDING BLOCK-4 ON BLOCK-2 (POS)
  MAKING STACK STACK-7 BLOCK-4 BLOCK-2
  (2) LETTING GO OF BLOCK-4
  (1) MOVING HAND FROM (1040 940 401) TO (990 650 101)
GOAL G-1 RETRY PUTON1 BLOCK-4 BLOCK-3
. GOAL G-22 CLEAROFF BLOCK-4
    G-22 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-3
  FOUNDSPACE DUPLICATED (600 840 301)
. GOAL G-23 CLEAROFF BLOCK-4
    G-23 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-3
  FOUNDSPACE DUPLICATED (600 840 301)
  G-1 EXHAUSTED
GOAL GT RETRY WITH PACK
GOAL G-24 CLEAROFF BLOCK-3
  G-24 SUCCEEDS
FOUND REGION CLEARTOP BLOCK-3
GOAL G-25 PUT BLOCK-4 (600 840 301)
. GOAL G-26 GRASP BLOCK-4
. . GOAL G-27 CLEAROFF BLOCK-4
      G-27 SUCCEEDS
    (1) MOVING HAND FROM (900 850 101) TO (1040 940 401)
    (2) GRASPING BLOCK-4
    G-26 SUCCEEDS
  (3) LIFTING BLOCK-4 FROM (940 840 201) TO (600 840 301)
  TAKING BLOCK-4 FROM STACK-7
  STACK-7 DISMANTLED
  (4) LETTING GO OF BLOCK-4
  MAKING STACK STACK-8 BLOCK-4 BLOCK-3
  ADDING BLOCK-4 ON BLOCK-3 (POS)
  G-25 SUCCEEDS
GOAL G-28 PUTON PYRAMID-1 ONTO BLOCK-4
. GOAL G-29 CLEAROFF PYRAMID-1
    G-29 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-4
. GOAL G-30 PUT PYRAMID-1 (600 840 501)
. . GOAL G-31 GRASP PYRAMID-1
. . . GOAL G-32 CLEAROFF PYRAMID-1
        G-32 SUCCEEDS
      (6) MOVING HAND FROM (700 940 501) TO (990 650 101)
      (7) GRASPING PYRAMID 1
      G-31 SUCCEEDS
    (8) LIFTING PYRAMID-1 FROM (940 600 1) TO (600 840 501)
    (9) LETTING GO OF PYRAMID-1
    ADDING PYRAMID-1 ON BLOCK-4 (POS)
    ADDING PYRAMID-1 TO STACK-8
    G-30 SUCCEEDS
  G-28 SUCCEEDS
REJECTING (629 920 301)
LOOKING AT (629 1040 301)
FOUND REGION (600 1040 301) TO (800 1140 301)
GOAL G-33 PUT BLOCK-1 (600 1040 301)
. GOAL G-34 GRASP BLOCK-1
. . GOAL G-35 CLEAROFF BLOCK-1
      G-35 SUCCEEDS
    (11) MOVING HAND FROM (650 890 601) TO (450 690 500)
    (12) GRASPING BLOCK-1
    G-34 SUCCEEDS
  (13) LIFTING BLOCK-1 FROM (400 640 400) TO (600 1040 301)
  TAKING BLOCK-1 FROM STACK-4
  (14) LETTING GO OF BLOCK-1
  ADDING BLOCK-1 TO STACK-8
  ADDING BLOCK-1 ON BLOCK-3 (POS)
  G-33 SUCCEEDS
GOAL G-36 PUTON PYRAMID-3 ONTO BLOCK-1
. GOAL G-37 CLEAROFF PYRAMID-3
    G-37 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-1
. GOAL G-38 PUT PYRAMID-3 (600 1040 401)
. . GOAL G-39 GRASP PYRAMID-3
. . . GOAL G-40 CLEAROFF PYRAMID-3
        G-40 SUCCEEDS
      (16) MOVING HAND FROM (650 1090 401) TO (550 690 640)
      (17) GRASPING PYRAMID-3
      G-39 SUCCEEDS
    (18) LIFTING PYRAMID-3 FROM (500 640 400) TO (600 1040 401)
    TAKING PYRAMID-3 FROM STACK-4
    STACK-4 DISMANTLED
    (19) LETTING GO OF PYRAMID-3
    ADDING PYRAMID-3 ON BLOCK-1 (POS)
    ADDING PYRAMID-3 TO STACK-8
    G-38 SUCCEEDS
  G-36 SUCCEEDS
GT SUCCEEDS

REPLY (1 (FAILED TO PUT PYRAMID-3 ON)) (1 (FAILED TO PUT PYRAMID-1 ON))
  (2 (OKAY))


...................................................................
.                                                                 .
.                                  SRP3++H1                       .
.                                  SRB1                           .
.                                  SGP1                           .
.                                  LGB4                           .
.                                  LRB3                LGB2       .
.                                                                 .
```

```
.                                                                 .
.                                                                 .
.            LBB5              ++X1LBP2                           .
.                                                                 .
.                                                                 .
.                                                                 .
.                                                                 .
.                                                                 .
.                                                                 .
.                                                                 .
.                                                                 .
.                                                                 .
.                                                                 .
...................................................................



13 INPUT TEXT IS " PUT THE BLUE BLOCK IN THE BOX "
OBJ-1 AMBIG B3-1 BLOCK-5 PYRAMID-2 ...
OBJ-1 REFERS BLOCK-5
OBJ-2 REFERS BOX-1
RELINCON OBJ-1 B4-1 IN BOX-1 POS
PUTIN STARTS WITH PUTON
STARTING GT PUTON BLOCK-5 ONTO BOX-1
GOAL G-1 CLEAROFF BLOCK-5
  G-1 SUCCEEDS
REJECTING (780 764 1)
LOOKING AT (780 840 1)
REGION AT (600 840 1) TOO SMALL
LOOKING AT (868 946 1)
REGION AT (800 840 1) TOO SMALL
REJECTING (742 706 1)
LOOKING AT (742 640 1)
REGION AT (600 600 1) TOO SMALL
REJECTING (692 682 1)
LOOKING AT (692 640 1)
REGION AT (600 600 1) TOO SMALL
LOOKING AT (841 899 1)
REGION AT (800 840 1) TOO SMALL
LOOKING AT (632 616 1)
REGION AT (600 600 1) TOO SMALL
LOOKING AT (800 1102 1)
FOUND REGION (800 1040 1) TO (1200 1200 1)
GOAL G-2 PUT BLOCK-5 (800 1040 1)
. GOAL G-3 GRASP BLOCK-5
. . GOAL G-4 CLEAROFF BLOCK-5
      G-4 SUCCEEDS
    (0) MOVING HAND FROM (650 1090 641) TO (450 890 400)
    (1) GRASPING BLOCK-5
    G-3 SUCCEEDS
  (2) LIFTING BLOCK-5 FROM (300 640 0) TO (800 1040 1)
  (3) LETTING GO OF BLOCK-5
  ADDING BLOCK-5 ON BOX-1 (POS)
  ADDING BLOCK-5 IN BOX-1 (POS)
  G-2 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))


...................................................................
.                                                                 .
.                                  SRP3               ++H1        .
.                                  SRB1      LBB5                 .
.                                  SGP1                           .
.                                  LGB4                           .
.                                  LRB3               LGB2        .
.                                                                 .
.                                                                 .
.                                                                 .
.                                  ++X1LBP2                       .
.                                                                 .
.                                                                 .
.
.
.
.
.
.
.
.
```

```
................................................................................
```

```
ADDING BLOCK BLOCK-9
ADDING BLOCK BLOCK-8
ADDING BLOCK BLOCK-7
ADDING BLOCK BLOCK-6
ADDING BLOCK-9 ON TABLE-1 (POS)
ADDING BLOCK-8 ON TABLE-1 (POS)
ADDING BLOCK-7 ON TABLE-1 (POS)
ADDING BLOCK-6 ON TABLE-1 (POS)
ADDING SIZE LARGE (POS) TO BLOCK-9
ADDING SIZE LARGE (POS) TO BLOCK-8
ADDING SIZE LARGE (POS) TO BLOCK-7
ADDING SIZE LARGE (POS) TO BLOCK-6
ADDING COLOR BLACK (POS) TO BLOCK-9
ADDING COLOR BLACK (POS) TO BLOCK-8
ADDING COLOR BLACK (POS) TO BLOCK-7
ADDING COLOR BLACK (POS) TO BLOCK-6



CLEARTOP (BLOCK-2) (BLOCK-5) (BLOCK-6) (BLOCK-7) (BLOCK-8) (BLOCK-9) (PYRAMID-1)
   (PYRAMID-2) (PYRAMID-3)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
   (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
   (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
   (BLOCK-5 SIZE LARGE POS) (BLOCK-6 COLOR BLACK POS) (BLOCK-6 SIZE LARGE POS)
   (BLOCK-7 COLOR BLACK POS) (BLOCK-7 SIZE LARGE POS) (BLOCK-8 COLOR BLACK POS)
   (BLOCK-8 SIZE LARGE POS) (BLOCK-9 COLOR BLACK POS) (BLOCK-9 SIZE LARGE POS)
   (PYRAMID-1 COLOR GREEN POS) (PYRAMID-1 SIZE SMALL POS)
   (PYRAMID-2 COLOR BLUE POS) (PYRAMID-2 SIZE LARGE POS)
   (PYRAMID-3 COLOR RED POS) (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-1 ON BLOCK-3 POS) (BLOCK-2 IN BOX-1 POS) (BLOCK-3 IN BOX-1 POS)
   (BLOCK-4 ON BLOCK-3 POS) (BLOCK-5 IN BOX-1 POS) (BLOCK-6 ON TABLE-1 POS)
   (BLOCK-7 ON TABLE-1 POS) (BLOCK-8 ON TABLE-1 POS) (BLOCK-9 ON TABLE-1 POS)
   (BOX-1 ON TABLE-1 POS) (PYRAMID-1 ON BLOCK-4 POS) (PYRAMID-2 IN BOX-1 POS)
   (PYRAMID-3 ON BLOCK-1 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
   (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BLOCK-6 200 200 200)
   (BLOCK-7 200 200 200) (BLOCK-8 200 200 200) (BLOCK-9 200 200 200)
   (BOX-1 600 600 1) (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200)
   (PYRAMID-3 100 100 240) (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-8) (BLOCK-3 STACK-8) (BLOCK-4 STACK-8)
   (PYRAMID-1 STACK-8) (PYRAMID-3 STACK-8)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
   (BLOCK-5 BLOCK) (BLOCK-6 BLOCK) (BLOCK-7 BLOCK) (BLOCK-8 BLOCK)
   (BLOCK-9 BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
   (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-1 600 1040 301) (BLOCK-2 940 840 1) (BLOCK-3 600 840 1)
   (BLOCK-4 600 840 301) (BLOCK-5 800 1040 1) (BLOCK-6 100 0 0) (BLOCK-7 400 0 0)
   (BLOCK-8 600 0 0) (BLOCK-9 900 0 0) (BOX-1 600 600 0) (HAND-1 950 1050 401)
   (PYRAMID-1 600 840 501) (PYRAMID-2 640 640 1) (PYRAMID-3 600 1040 401)
   (TABLE-1 0 0 0)
```

```
................................................................................
.                                                                              .
.                                             SRP3                **H1         .
.                                             SRB1       LBB5                  .
.                                             SGP1                             .
.                                             LGB4                             .
.                                             LRB3                LGB2         .
.                                                                              .
.                                                                              .
.                                                                              .
.                                             **X1LBP2                         .
.                                                                              .
.                                                                              .
.                                                                              .
.                                                                              .
.                                                                              .
.                                                                              .
.                                                                              .
.     LBB6                LBB7                LBB8                LBB9         .
................................................................................
```

```
RUN TIME 11 MIN. 14.9 SEC

EXAM    TRY     FIRE    UNACT   E/F     E/T     T/F
5383    1540    942     3400    5.71    3.40    1.64
0.125   0.426   0.716   0.190   SEC AVG

2017 INSERTS 1391 DELETES 519 WARNINGS 19 NEW OBJECTS
MAX :SMPX LENGTH 194
CORE (FREE.FULL): (14376 . 2900) USED (500 . 64)
FIRED 4 OUT OF 440 PRODS


                 - - - - - - - - - - - - - - - - - -


FIFTH SEGMENT

15 INPUT TEXT IS " PUT A BLACK BLOCK ON THE LARGE RED BLOCK "
OBJ-1 AMBIG 83-1 BLOCK-6 BLOCK-7 ...
OBJ-2 AMBIG L7-1 BLOCK-2 BLOCK-3 ...
OBJ-2 REFERS BLOCK-3
CHOOSING BLOCK-9 FOR OBJ-1
RELINCON OBJ-1 84-1 ON BLOCK-3 POS
STARTING GT PUTON BLOCK-9 ONTO BLOCK-3
GOAL G-1 CLEAROFF BLOCK-9
  G-1 SUCCEEDS
REJECTING (605 909 301)
LOOKING AT (605 1040 301)
REGION AT (600 1040 301) TOO SMALL
REJECTING (600 843 301)
LOOKING AT (600 1040 301)
REGION AT (600 1040 301) TOO SMALL
REJECTING (665 930 301)
LOOKING AT (665 1040 301)
REGION AT (600 1040 301) TOO SMALL
REJECTING (627 953 301)
LOOKING AT (627 1040 301)
REGION AT (600 1040 301) TOO SMALL
REJECTING (647 993 301)
LOOKING AT (647 1040 301)
REGION AT (600 1040 301) TOO SMALL
FINDSPACE LIMIT EXCEEDED
NO SPACE TO PUTON BLOCK-9 BLOCK-3
GT FAILS
GOAL G-2 MAKESPACE FOR BLOCK-9 ON BLOCK-3
. GOAL G-3 GETRIDOF BLOCK-4
  REJECTING (794 42 0)
  LOOKING AT (800 42 0)
  REGION AT (800 0 0) TOO SMALL
  REJECTING (295 156 0)
  LOOKING AT (300 156 0)
  REGION AT (300 0 0) TOO SMALL
  LOOKING AT (219 651 0)
  FOUND REGION (0 200 0) TO (600 600 0)
. . GOAL G-4 PUT BLOCK-4 (354 209 0)
. . . GOAL G-5 GRASP BLOCK-4
. . . . GOAL G-6 CLEAROFF BLOCK-4
. . . . . GOAL G-7 GETRIDOF PYRAMID-1
          REJECTING (200 57 0)
          LOOKING AT (100 57 0)
          FOUND REGION (0 0 0) TO (100 600 0)
. . . . . . GOAL G-8 PUT PYRAMID-1 (0 57 0)
. . . . . . . GOAL G-9 GRASP PYRAMID-1
. . . . . . . . GOAL G-10 CLEAROFF PYRAMID-1
                  G-10 SUCCEEDS
                (0) MOVING HAND FROM (950 1050 401) TO (600 840 801)
                (1) GRASPING PYRAMID-1
                G-9 SUCCEEDS
              (2) LIFTING PYRAMID-1 FROM (600 840 501) TO (0 57 0)
              TAKING PYRAMID-1 FROM STACK-8
              (3) LETTING GO OF PYRAMID-1
              ADDING PYRAMID-1 ON TABLE-1 (POS)
              G-8 SUCCEEDS
            G-7 SUCCEEDS
          G-6 SUCCEEDS
        (4) MOVING HAND FROM (50 107 100) TO (700 940 501)
        (5) GRASPING BLOCK-4
        G-5 SUCCEEDS
      (6) LIFTING BLOCK-4 FROM (600 840 301) TO (354 200 0)
      TAKING BLOCK-4 FROM STACK-8
      (7) LETTING GO OF BLOCK-4
      ADDING BLOCK-4 ON TABLE-1 (POS)
      G-4 SUCCEEDS
```

```
    G-3 SUCCEEDS
  LOOKING AT (616 989 301)
  FOUND REGION (600 840 301) TO (800 1040 301)
  G-2 SUCCEEDS
GOAL G-11 PUT BLOCK-9 (600 840 301)
. GOAL G-12 GRASP BLOCK-9
. . GOAL G-13 CLEAROFF BLOCK-9
      G-13 SUCCEEDS
    (8) MOVING HAND FROM (454 309 200) TO (1000 100 200)
    (9) GRASPING BLOCK-9
    G-12 SUCCEEDS
  (10) LIFTING BLOCK-9 FROM (900 0 0) TO (600 840 301)
  (11) LETTING GO OF BLOCK-9
  ADDING BLOCK-9 ON BLOCK-3 (POS)
  ADDING BLOCK-9 TO STACK-8
  G-11 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))
```

```
.......................................................................
.                                                                     .
.                                  SPP3                               .
.                                  SRB1         LBB5                  .
.                                                                     .
.                                  LBB9  ++H1                         .
.                                  LRB3                    LGB2       .
.                                                                     .
.                                                                     .
.                                  ++X1LBP2                           .
.                                                                     .
.                                                                     .
.                                                                     .
.                   LGB4                                              .
.                                                                     .
SGP1  LBB6                  LBB7         LBB8
.......................................................................
```

```
16 INPUT TEXT IS " PUT A LARGE GREEN BLOCK IN THE BOX "
OBJ-1 AMBIG L3-1 BLOCK-2 BLOCK-3 ...
OBJ-1 AMBIG G4-1 BLOCK-2 BLOCK-4 ...
OBJ-2 REFERS BOX-1
RELRESTR OBJ-1 BS-1 IN BOX-1 POS
OBJ-1 REFERS BLOCK-2
BACKUP OBJ-1 REFERS BLOCK-4
PUTIN STARTS WITH PUTON
STARTING GT PUTON BLOCK-4 ONTO BOX-1
GOAL G-1 CLEAROFF BLOCK-4
  G-1 SUCCEEDS
LOOKING AT (870 913 1)
REGION AT (800 840 1) TOO SMALL
LOOKING AT (870 844 1)
REGION AT (800 840 1) TOO SMALL
LOOKING AT ()870 810 1)
FOUND REGION (940 600 1) TO (1200 840 1)
GOAL G-2 PUT BLOCK-4 (940 600 1)
. GOAL G-3 GRASP BLOCK-4
. . GOAL G-4 CLEAROFF BLOCK-4
      G-4 SUCCEEDS
    (0) MOVING HAND FROM (700 940 501) TO (454 309 200)
    (1) GRASPING BLOCK-4
    G-3 SUCCEEDS
  (2) LIFTING BLOCK-4 FROM (354 209 0) TO (940 600 1)
  ADDING BLOCK-4 ON BOX-1 (POS)
  (3) LETTING GO OF BLOCK-4
  ADDING BLOCK-4 IN BOX-1 (POS)
  G-2 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))

.......................................................................
```

```
.                                                                     .
.                                  SPP3                               .
.                                  SRB1         LBB5                  .
.                                                                     .
.                                  LBB9                               .
.                                  LRB3                     LGB2      .
.                                                                     .
.                                                        ++H1         .
.                                  ++X1LBP2                 LGB4      .
.                                                                     .
.                                                                     .
.                                                                     .
SGP1  LBB6                  LBB7        LBB8                          .
.......................................................................
```

```
17 INPUT TEXT IS " PICK A BLACK BLOCK UP "
OBJ-1 AMBIG B3-1 BLOCK-6 BLOCK-7 ...
CHOOSING BLOCK-9 FOR OBJ-1
STARTING GT PICKUP BLOCK-9
GOAL G-1 GRASP BLOCK-9
. GOAL G-2 CLEAROFF BLOCK-9
    G-2 SUCCEEDS
  (0) MOVING HAND FROM (1040 700 201) TO (700 940 501)
  (1) GRASPING BLOCK-9
  G-1 SUCCEEDS
(2) LIFTING BLOCK-9 FROM (600 840 301) TO (600 840 1000)
TAKING BLOCK-9 FROM STACK-8
GT SUCCEEDS

REPLY (1 (OKAY))
```

```
CLEARTOP (BLOCK-2) (BLOCK-4) (BLOCK-5) (BLOCK-6) (BLOCK-7) (BLOCK-8) (BLOCK-9)
  (PYRAMID-1) (PYRAMID-2) (PYRAMID-3)
GRASPING (HAND-1 BLOCK-9)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
  (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
  (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
  (BLOCK-5 SIZE LARGE POS) (BLOCK-6 COLOR BLACK POS) (BLOCK-6 SIZE LARGE POS)
  (BLOCK-7 COLOR BLACK POS) (BLOCK-7 SIZE LARGE POS) (BLOCK-8 COLOR BLACK POS)
  (BLOCK-8 SIZE LARGE POS) (BLOCK-9 COLOR BLACK POS) (BLOCK-9 SIZE LARGE POS)
  (PYRAMID-1 COLOR GREEN POS) (PYRAMID-1 SIZE SMALL POS)
  (PYRAMID-2 COLOR BLUE POS) (PYRAMID-2 SIZE LARGE POS)
  (PYRAMID-3 COLOR RED POS) (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-1 ON BLOCK-3 POS) (BLOCK-2 IN BOX-1 POS) (BLOCK-3 IN BOX-1 POS)
  (BLOCK-4 IN BOX-1 POS) (BLOCK-5 IN BOX-1 POS) (BLOCK-6 ON TABLE-1 POS)
  (BLOCK-7 ON TABLE-1 POS) (BLOCK-8 ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS)
  (PYRAMID-1 ON TABLE-1 POS) (PYRAMID-2 IN BOX-1 POS) (PYRAMID-3 ON BLOCK-1 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
  (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BLOCK-6 200 200 200)
  (BLOCK-7 200 200 200) (BLOCK-8 200 200 200) (BLOCK-9 200 200 200)
  (BOX-1 600 600 1) (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200)
  (PYRAMID-3 100 100 240) (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-8) (BLOCK-3 STACK-8) (PYRAMID-3 STACK-8)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
  (BLOCK-5 BLOCK) (BLOCK-6 BLOCK) (BLOCK-7 BLOCK) (BLOCK-8 BLOCK)
  (BLOCK-9 BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
  (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-1 600 1040 301) (BLOCK-2 940 840 1) (BLOCK-3 600 840 1)
  (BLOCK-4 940 600 1) (BLOCK-5 800 1040 1) (BLOCK-6 100 0 0) (BLOCK-7 400 0 0)
  (BLOCK-8 600 0 0) (BLOCK-9 600 840 1000) (BOX-1 600 600 0)
  (HAND-1 700 940 1200) (PYRAMID-1 0 57 0) (PYRAMID-2 640 640 1)
  (PYRAMID-3 600 1040 401) (TABLE-1 0 0 0)
```

```
.......................................................................
.                                                                     .
.                                  SPP3                               .
.                                  SRB1         LBB5                  .
.                                                                     .
.                                  LBB9  ++H1                         .
.                                  LRB3                     LGB2
```

```
.
.
.
.                                          **XILBP2              LG04
.
.
.
.
.
.
.
.
.
SGP1  L006                  L007           L000
.......................................................................



RUN TIME 5 MIN. 32.3 SEC

EXAM      TRY       FIPE      WMACT   E/F     E/T     T/F
3323      767       547       1810    6.07    4.33    1.40
P 1R8     0.433     0.000     0.104   SEC AVG

1186 INSERTS 724 DELETES 317 WARNINGS 18 NEW OBJECTS
MAX :SMPX LENGTH 172
CORE (FREE.FULL): (14311 . 2010) USED (1464 . 164)
FIRED 44 OUT OF 408 PRODS


                    - - - - - - - - - - - - - - - - - -


SIXTH SEGMENT

10.0 INPUT TEXT IS " PUT IT IN THE BOX "
OBJ-1 REFERS BLOCK-9
OBJ-2 REFERS BOX-1
RELINCON OBJ-1 I2-1 IN BOX-1 POS
PUTIN STARTS WITH PUTON
STARTING GT PUTON BLOCK-9 ONTO BOX-1
GOAL G-1 CLEAROFF BLOCK-9
  G-1 SUCCEEDS
LOOKING AT (699 631 1)
REGION AT (600 600 1) TOO SMALL
REJECTING (801 830 1)
LOOKING AT (801 840 1)
REGION AT (800 840 1) TOO SMALL
REJECTING (835 747 1)
LOOKING AT (835 840 1)
REGION AT (800 840 1) TOO SMALL
LOOKING AT (892 1027 1)
REGION AT (800 840 1) TOO SMALL
LOOKING AT (827 614 1)
REGION AT (800 600 1) TOO SMALL
REJECTING (604 1060 1)
LOOKING AT (604 1140 1)
REGION AT (600 1140 1) TOO SMALL
LOOKING AT (814 845 1)
REGION AT (800 840 1) TOO SMALL
FINDSPACE LIMIT EXCEEDED
NO SPACE TO PUTON BLOCK-9 BOX-1
GT FAILS
GOAL G-2 CLEAROFF BOX-1
. GOAL G-3 GETRIDOF BLOCK-5
  REJECTING (455 139 0)
  LOOKING AT (400 139 0)
  REGION AT (300 0 0) TOO SMALL
  REJECTING (903 929 0)
  LOOKING AT (903 600 0)
  FOUND REGION (800 200 0) TO (1200 600 0)
. . GOAL G-4 PUT BLOCK-5 (864 267 0)
. . . GOAL G-5 GRASP BLOCK-5
. . . . GOAL G-6 GETRIDOF BLOCK-9
        REJECTING (702 124 0)
        LOOKING AT (702 200 0)
        FOUND REGION (600 200 0) TO (1200 600 0)
. . . . . GOAL G-7 PUT BLOCK-9 (906 307 0)
. . . . . . GOAL G-8 GRASP BLOCK-9
              G-8 SUCCEEDS
            (0) LIFTING BLOCK-9 FROM (600 840 1000) TO (906 307 0)
            (1) LETTING GO OF BLOCK-9
```

```
                ADDING BLOCK-9 ON TABLE-1 (POS)
                G-7 SUCCEEDS
              G-6 SUCCEEDS
. . . . GOAL G-9 CLEAROFF BLOCK-5
              G-9 SUCCEEDS
            (2) MOVING HAND FROM (1005 407 200) TO (900 1000 401)
            (3) GRASPING BLOCK-5
            G-5 SUCCEEDS
          MOVE TO (1014 317 400) OVERLAPS BLOCK-5 WITH BLOCK-9
          (4) LETTING GO OF BLOCK-5
          G-4 SUCCEEDS
        G-3 SUCCEEDS
. GOAL G-10 GETRIDOF BLOCK-5
  REJECTING (755 1075 0)
  LOOKING AT (600 1075 0)
  REGION AT (600 507 0) TOO SMALL
  REJECTING (940 893 0)
  LOOKING AT (940 600 0)
  REGION AT (800 507 0) TOO SMALL
  LOOKING AT (260 432 0)
  FOUND REGION (100 200 0) TO (600 307 0)
. . GOAL G-11 PUT BLOCK-5 (253 203 0)
. . . GOAL G-12 GRASP BLOCK-5
. . . . GOAL G-13 CLEAROFF BLOCK-5
          G-13 SUCCEEDS
        (5) GRASPING BLOCK-5
        G-12 SUCCEEDS
      (6) LIFTING BLOCK-5 FROM (800 1040 1) TO (253 203 0)
      (7) LETTING GO OF BLOCK-5
      ADDING BLOCK-5 ON TABLE-1 (POS)
      G-11 SUCCEEDS
    G-10 SUCCEEDS
. GOAL G-14 GETRIDOF BLOCK-4
  REJECTING (784 35 0)
  LOOKING AT (800 35 0)
  FOUND REGION (800 0 0) TO (1200 307 0)
. . GOAL G-15 PUT BLOCK-4 (983 66 0)
. . . GOAL G-16 GRASP BLOCK-4
. . . . GOAL G-17 CLEAROFF BLOCK-4
          G-17 SUCCEEDS
        (8) MOVING HAND FROM (403 253 400) TO (1040 700 201)
        (9) GRASPING BLOCK-4
        G-16 SUCCEEDS
      (10) LIFTING BLOCK-4 FROM (940 600 1) TO (983 66 0)
      (11) LETTING GO OF BLOCK-4
      ADDING BLOCK-4 ON TABLE-1 (POS)
      G-15 SUCCEEDS
    G-14 SUCCEEDS
. GOAL G-18 GETRIDOF BLOCK-2
  LOOKING AT (181 432 0)
  FOUND REGION (100 303 0) TO (600 800 0)
. . GOAL G-19 PUT BLOCK-2 (197 351 0)
. . . GOAL G-20 GRASP BLOCK-2
. . . . GOAL G-21 CLEAROFF BLOCK-2
          G-21 SUCCEEDS
        (12) MOVING HAND FROM (1083 166 200) TO (1040 940 201)
        (13) GRASPING BLOCK-2
        G-20 SUCCEEDS
      (14) LIFTING BLOCK-2 FROM (940 840 1) TO (197 351 0)
      (15) LETTING GO OF BLOCK-2
      ADDING BLOCK-2 ON TABLE-1 (POS)
      G-19 SUCCEEDS
    G-18 SUCCEEDS
. GOAL G-22 GETRIDOF PYRAMID-2
  LOOKING AT (79 475 0)
  REGION AT (0 303 0) TOO SMALL
  LOOKING AT (370 730 0)
  FOUND REGION (300 551 0) TO (600 1200 0)
. . GOAL G-23 PUT PYRAMID-2 (300 803 0)
. . . GOAL G-24 GRASP PYRAMID-2
. . . . GOAL G-25 CLEAROFF PYRAMID-2
          G-25 SUCCEEDS
        (16) MOVING HAND FROM (297 451 200) TO (780 740 201)
        (17) GRASPING PYRAMID-2
        G-24 SUCCEEDS
      (18) LIFTING PYRAMID-2 FROM (840 640 1) TO (300 803 0)
      (19) LETTING GO OF PYRAMID-2
      ADDING PYRAMID-2 ON TABLE-1 (POS)
      G-23 SUCCEEDS
    G-22 SUCCEEDS
. GOAL G-26 GETRIDOF BLOCK-3
  LOOKING AT (1023 522 0)
  REGION AT (800 507 0) TOO SMALL
  LOOKING AT (40 610 0)
  FOUND REGION (0 551 0) TO (300 1200 0)
. . GOAL G-27 PUT BLOCK-3 (41 700 0)
. . . GOAL G-28 GRASP BLOCK-3
. . . . GOAL G-29 CLEAROFF BLOCK-3
. . . . . GOAL G-30 GETRIDOF BLOCK-1
            REJECTING (1095 772 0)
            LOOKING AT (1095 800 0)
            REGION AT (800 551 0) TOO SMALL
            LOOKING AT (416 1065 0)
            FOUND REGION (397 1003 0) TO (600 1200 0)
. . . . . . . GOAL G-31 PUT BLOCK-1 (457 1022 0)
. . . . . . . . GOAL G-32 GRASP BLOCK-1
. . . . . . . . . GOAL G-33 CLEAROFF BLOCK-1
. . . . . . . . . . GOAL G-34 GETRIDOF PYRAMID-3
                      LOOKING AT (577 305 0)
                      FOUND REGION (553 303 0) TO (905 600 0)
. . . . . . . . . . . GOAL G-35 PUT PYRAMID-3 (797 396 0)
. . . . . . . . . . . . GOAL G-36 GRASP PYRAMID-3
. . . . . . . . . . . . . GOAL G-37 CLEAROFF PYRAMID-3
                              G-37 SUCCEEDS
                             (20) MOVING HAND FROM (450 903 200) TO (650 1090 641)
                             (21) GRASPING PYRAMID-3
                             G-36 SUCCEEDS
                            (22) LIFTING PYRAMID-3 FROM (600 1040 401) TO (797 396 0)
                            TAKING PYRAMID-3 FROM STACK-8
                            ADDING PYRAMID-3 ON TABLE-1 (POS)
                            (23) LETTING GO OF PYRAMID-3
                            G-35 SUCCEEDS
                          G-34 SUCCEEDS
                        G-33 SUCCEEDS
                      (24) MOVING HAND FROM (847 446 240) TO (650 1090 401)
                      (25) GRASPING BLOCK-1
                      G-32 SUCCEEDS
                    (26) LIFTING BLOCK-1 FROM (600 1040 301) TO (457 1022 0)
                    TAKING BLOCK-1 FROM STACK-8
                    STACK-8 DISMANTLED
                    (27) LETTING GO OF BLOCK-1
                    ADDING BLOCK-1 ON TABLE-1 (POS)
                    G-31 SUCCEEDS
                  G-30 SUCCEEDS
                G-29 SUCCEEDS
              (28) MOVING HAND FROM (507 1072 100) TO (700 900 301)
              (29) GRASPING BLOCK-3
              G-28 SUCCEEDS
            (30) LIFTING BLOCK-3 FROM (600 840 1) TO (41 700 0)
            (31) LETTING GO OF BLOCK-3
            ADDING BLOCK-3 ON TABLE-1 (POS)
            G-27 SUCCEEDS
          G-26 SUCCEEDS
        G-2 SUCCEEDS
FOUND REGION CLEARTOP BOX-1
GOAL G-38 PUT BLOCK-3 (600 800 1)
. GOAL G-39 GRASP BLOCK-3
. . GOAL G-40 CLEAROFF BLOCK-3
      G-40 SUCCEEDS
    (33) GRASPING BLOCK-3
    G-39 SUCCEEDS
  (34) LIFTING BLOCK-3 FROM (41 700 0) TO (600 800 1)
  (35) LETTING GO OF BLOCK-3
  ADDING BLOCK-3 ON BOX-1 (POS)
  ADDING BLOCK-3 IN BOX-1 (POS)
  G-38 SUCCEEDS
GOAL G-41 PUTON PYRAMID-3 ONTO BLOCK-3
. GOAL G-42 CLEAROFF PYRAMID-3
    G-42 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-3
. GOAL G-43 PUT PYRAMID-3 (650 700 301)
. . GOAL G-44 GRASP PYRAMID-3
. . . GOAL G-45 CLEAROFF PYRAMID-3
        G-45 SUCCEEDS
      (37) MOVING HAND FROM (700 750 301) TO (847 446 240)
      (38) GRASPING PYRAMID-3
      G-44 SUCCEEDS
    (39) LIFTING PYRAMID-3 FROM (797 396 0) TO (650 700 301)
    (40) LETTING GO OF PYRAMID-3
    ADDING PYRAMID-3 ON BLOCK-3 (POS)
    MAKING STACK STACK-9 PYRAMID-3 BLOCK-3
    G-43 SUCCEEDS
  G-41 SUCCEEDS
REJECTING (663 829 1)
LOOKING AT (663 900 1)
FOUND REGION (600 900 1) TO (1200 1200 1)
```

```
GOAL G-46 PUT PYRAMID-2 (800 900 1)
. GOAL G-47 GRASP PYRAMID-2
. . GOAL G-48 CLEAROFF PYRAMID-2
      G-48 SUCCEEDS
    (42) MOVING HAND FROM (700 750 541) TO (450 903 200)
    (43) GRASPING PYRAMID-2
    G-47 SUCCEEDS
  (44) LIFTING PYRAMID-2 FROM (300 803 0) TO (800 900 1)
  (45) LETTING GO OF PYRAMID-2
  ADDING PYRAMID-2 ON BOX-1 (POS)
  ADDING PYRAMID-2 IN BOX-1 (POS)
  G-46 SUCCEEDS
REJECTING (664 737 1)
LOOKING AT (800 737 1)
FOUND REGION (800 600 1) TO (1200 900 1)
GOAL G-49 PUT BLOCK-2 (800 600 1)
. GOAL G-50 GRASP BLOCK-2
. . GOAL G-51 CLEAROFF BLOCK-2
      G-51 SUCCEEDS
    (47) MOVING HAND FROM (750 1000 201) TO (297 451 200)
    (48) GRASPING BLOCK-2
    G-50 SUCCEEDS
  (49) LIFTING BLOCK-2 FROM (197 351 0) TO (800 600 1)
  (50) LETTING GO OF BLOCK-2
  ADDING BLOCK-2 ON BOX-1 (POS)
  ADDING BLOCK-2 IN BOX-1 (POS)
  G-49 SUCCEEDS
GOAL G-52 PUTON BLOCK-4 ONTO BLOCK-2
. GOAL G-53 CLEAROFF BLOCK-4
    G-53 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-2
. GOAL G-54 PUT BLOCK-4 (800 600 201)
. . GOAL G-55 GRASP BLOCK-4
. . . GOAL G-56 CLEAROFF BLOCK-4
        G-56 SUCCEEDS
      (52) MOVING HAND FROM (900 700 201) TO (1003 106 200)
      (53) GRASPING BLOCK-4
      G-55 SUCCEEDS
    (54) LIFTING BLOCK-4 FROM (903 66 0) TO (800 600 201)
    (55) LETTING GO OF BLOCK-4
    ADDING BLOCK-4 ON BLOCK-2 (POS)
    MAKING STACK STACK-10 BLOCK-4 BLOCK-2
    G-54 SUCCEEDS
  G-52 SUCCEEDS
REJECTING (662 919 1)
LOOKING AT (662 900 1)
REGION AT (600 900 1) TOO SMALL
REJECTING (682 829 1)
LOOKING AT (682 900 1)
REGION AT (600 900 1) TOO SMALL
REJECTING (752 693 1)
LOOKING AT (800 693 1)
REGION AT (800 600 1) TOO SMALL
REJECTING (949 734 1)
LOOKING AT (1000 734 1)
REGION AT (1000 600 1) TOO SMALL
LOOKING AT (916 972 1)
FOUND REGION (900 900 1) TO (1200 1200 1)
GOAL G-57 PUT BLOCK-5 (900 900 1)
. GOAL G-58 GRASP BLOCK-5
. . GOAL G-59 CLEAROFF BLOCK-5
      G-59 SUCCEEDS
    (57) MOVING HAND FROM (900 700 401) TO (403 253 400)
    (58) GRASPING BLOCK-5
    G-58 SUCCEEDS
  (59) LIFTING BLOCK-5 FROM (253 203 0) TO (900 900 1)
  (60) LETTING GO OF BLOCK-5
  ADDING BLOCK-5 ON BOX-1 (POS)
  ADDING BLOCK-5 IN BOX-1 (POS)
  G-57 SUCCEEDS
GOAL G-60 PUTON BLOCK-1 ONTO BLOCK-5
. GOAL G-61 CLEAROFF BLOCK-1
    G-61 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-5
. GOAL G-62 PUT BLOCK-1 (1000 900 401)
. . GOAL G-63 GRASP BLOCK-1
. . . GOAL G-64 CLEAROFF BLOCK-1
        G-64 SUCCEEDS
      (62) MOVING HAND FROM (1050 550 401) TO (507 1072 100)
      (63) GRASPING BLOCK-1
      G-63 SUCCEEDS
    (64) LIFTING BLOCK-1 FROM (457 1022 0) TO (1000 900 401)
    (65) LETTING GO OF BLOCK-1
    ADDING BLOCK-1 ON BLOCK-5 (POS)
    MAKING STACK STACK-11 BLOCK-1 BLOCK-5
    G-62 SUCCEEDS
  G-60 SUCCEEDS
REJECTING (854 1053 1)
LOOKING AT (900 1053 1)
FOUND REGION (900 1000 1) TO (1200 1200 1)
GOAL G-65 PUT BLOCK-9 (900 1000 1)
. GOAL G-66 GRASP BLOCK-9
. . GOAL G-67 CLEAROFF BLOCK-9
      G-67 SUCCEEDS
    (67) MOVING HAND FROM (1050 950 501) TO (1085 407 200)
    (68) GRASPING BLOCK-9
    G-66 SUCCEEDS
  (69) LIFTING BLOCK-9 FROM (985 307 0) TO (900 1000 1)
  (70) LETTING GO OF BLOCK-9
  ADDING BLOCK-9 ON BOX-1 (POS)
  ADDING BLOCK-9 IN BOX-1 (POS)
  G-66 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))
```

```
......................................................................
.
.                                                        **H1
.
.                                                   LB09
.                                       LBP2        LB05  SRB1
.
.
.
.                                       LRB3SRP3    LGB4
.                                       **X1        LGB2
.
.
.
.
.
.
.
.
SGP1  LB06                 LB07         LB08
......................................................................
```

```
10.5 INPUT TEXT IS " PICK A BLACK BLOCK ON THE TABLE UP "
OBJ-1 AMBIG B3-1 BLOCK-6 BLOCK-7 ...
OBJ-2 REFERS TABLE-1
RELRESTR OBJ-1 B4-1 ON TABLE-1 POS
OBJ-1 AMBIG B4-1 BLOCK-6 BLOCK-7 ...
CHOOSING BLOCK-8 FOR OBJ-1
STARTING GT PICKUP BLOCK-8
GOAL G-1 GRASP BLOCK-8
. GOAL G-2 CLEAROFF BLOCK-8
    G-2 SUCCEEDS
  (0) MOVING HAND FROM (1000 1100 201) TO (700 100 200)
  (1) GRASPING BLOCK-8
  G-1 SUCCEEDS
(2) LIFTING BLOCK-8 FROM (800 0 0) TO (800 0 1000)
GT SUCCEEDS

REPLY (1 (OKAY))
```

```
......................................................................
.
.
.
.                                                   LB09
.                                       LBP2        LB05  SRB1
.
.
.
.                                       LRB3SRP3    LGB4
.                                       **X1        LGB2
.
.
```

```
.
.
.
.
.
.
.
.                                                   **H1
SGP1  LB06                 LB07         LB08
......................................................................
```

```
10.6 INPUT TEXT IS " PUT IT IN THE BOX "
OBJ-1 REFERS BLOCK-8
OBJ-2 REFERS BOX-1
RELINCON OBJ-1 I2-1 IN BOX-1 POS
PUTIN STARTS WITH PUTON
STARTING GT PUTON BLOCK-8 ONTO BOX-1
GOAL G-1 CLEAROFF BLOCK-8
  G-1 SUCCEEDS
LOOKING AT (1052 896 1)
REGION AT (1000 800 1) TOO SMALL
REJECTING (849 1054 1)
LOOKING AT (849 1100 1)
REGION AT (800 1100 1) TOO SMALL
REJECTING (1031 936 1)
LOOKING AT (1031 900 1)
REGION AT (1000 900 1) TOO SMALL
REJECTING (610 762 1)
LOOKING AT (610 900 1)
REGION AT (600 900 1) TOO SMALL
LOOKING AT (1061 741 1)
FOUND REGION (1000 600 1) TO (1200 900 1)
GOAL G-2 PUT BLOCK-8 (1000 800 1)
. GOAL G-3 GRASP BLOCK-8
    G-3 SUCCEEDS
  (0) LIFTING BLOCK-8 FROM (600 0 1000) TO (1000 800 1)
  ADDING BLOCK-8 ON BOX-1 (POS)
  (1) LETTING GO OF BLOCK-8
  ADDING BLOCK-8 IN BOX-1 (POS)
  G-2 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))
```

```
CLEARTOP (BLOCK-1) (BLOCK-4) (BLOCK-6) (BLOCK-7) (BLOCK-8) (BLOCK-9) (PYRAMID-1)
  (PYRAMID-2) (PYRAMID-3)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
  (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
  (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
  (BLOCK-5 SIZE LARGE POS) (BLOCK-6 COLOR BLACK POS) (BLOCK-6 SIZE LARGE POS)
  (BLOCK-7 COLOR BLACK POS) (BLOCK-7 SIZE LARGE POS) (BLOCK-8 COLOR BLACK POS)
  (BLOCK-8 SIZE LARGE POS) (BLOCK-9 COLOR BLACK POS) (BLOCK-9 SIZE LARGE POS)
  (PYRAMID-1 COLOR GREEN POS) (PYRAMID-1 SIZE SMALL POS)
  (PYRAMID-2 COLOR BLUE POS) (PYRAMID-2 SIZE LARGE POS)
  (PYRAMID-3 COLOR RED POS) (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-1 ON BLOCK-5 POS) (BLOCK-2 IN BOX-1 POS) (BLOCK-3 IN BOX-1 POS)
  (BLOCK-4 ON BLOCK-2 POS) (BLOCK-5 IN BOX-1 POS) (BLOCK-6 ON TABLE-1 POS)
  (BLOCK-7 ON TABLE-1 POS) (BLOCK-8 IN BOX-1 POS) (BLOCK-9 IN BOX-1 POS)
  (BOX-1 ON TABLE-1 POS) (PYRAMID-1 ON TABLE-1 POS) (PYRAMID-2 IN BOX-1 POS)
  (PYRAMID-3 ON BLOCK-3 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
  (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BLOCK-6 200 200 200)
  (BLOCK-7 200 200 200) (BLOCK-8 200 200 200) (BLOCK-9 200 200 200)
  (BOX-1 600 600 1) (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200)
  (PYRAMID-3 100 100 240) (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-11) (BLOCK-2 STACK-10) (BLOCK-3 STACK-9)
  (BLOCK-4 STACK-10) (BLOCK-5 STACK-11) (PYRAMID-3 STACK-9)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
  (BLOCK-5 BLOCK) (BLOCK-6 BLOCK) (BLOCK-7 BLOCK) (BLOCK-8 BLOCK)
  (BLOCK-9 BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
  (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-1 1000 900 401) (BLOCK-2 800 600 1) (BLOCK-3 600 600 1)
  (BLOCK-4 800 600 201) (BLOCK-5 900 900 1) (BLOCK-6 100 0 0) (BLOCK-7 400 0 0)
  (BLOCK-8 1000 600 1) (BLOCK-9 900 1000 1) (BOX-1 600 600 0)
  (HAND-1 1100 700 201) (PYRAMID-1 0 57 0) (PYRAMID-2 600 900 1)
  (PYRAMID-3 650 700 301)) (TABLE-1 0 0 0)
```

```
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                                                 LB08                .
.                                            LBP2                 LB05  SMB1          .
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                            LRUSGRP3    LG04                 ++H1    .
.                                            ++X1        LG02           LB00          .
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                                                                     .
.                                                                                     .
SGP1  LB06                    LB07                                                    .
.......................................................................................
```

```
RUN TIME 10 MIN. 30.9 SEC

EXAM    TRY     FIRE    WMACT   E/F     E/T     T/F
6048    1644    1034    3905    5.85    3.68    1.59
0.185   0.681   1.08    0.287   SEC AVG

2317 INSERTS 1500 DELETES 613 WARNINGS 21 NEW OBJECTS
MAX :SMPX LENGTH 215
CORE (FREE.FULL): (15786 . 2047) USED (1987 . 271)
FIRED 88 OUT OF 407 PRODS
```

- -

SEVENTH SEGMENT

```
ADDING SIZE LARGE (POS) TO BLOCK-A
ADDING BLOCK BLOCK-A
19 INPUT TEXT IS " STACK UP A LARGE RED BLOCK AND A SMALL BLOCK AND IT AND A
  SMALL PYRAMID AND A BLACK BLOCK AND A LARGE GREEN BLOCK AND A SMALL PYRAMID "
OBJ-1 AMBIG L4-1 BLOCK-2 BLOCK-3 ...
OBJ-1 REFERS BLOCK-3
OBJ-2 AMBIG S9-1 BLOCK-1 PYRAMID-1 ...
OBJ-2 REFERS BLOCK-1
OBJ-3 REFERS BLOCK-A
OBJ-4 AMBIG S15-1 BLOCK-1 PYRAMID-1 ...
OBJ-4 AMBIG P16-1 PYRAMID-1 PYRAMID-3 ...
CHOOSING PYRAMID-3 FOR OBJ-4
OBJ-5 AMBIG B19-1 BLOCK-6 BLOCK-7 ...
CHOOSING BLOCK-9 FOR OBJ-5
OBJ-6 AMBIG L23-1 BLOCK-2 BLOCK-3 ...
OBJ-6 AMBIG G24-1 BLOCK-2 BLOCK-4 ...
CHOOSING BLOCK-4 FOR OBJ-6
OBJ-7 AMBIG S28-1 BLOCK-1 PYRAMID-1 ...
OBJ-7 AMBIG P29-1 PYRAMID-1 PYRAMID-3 ...
CHOOSING PYRAMID-1 FOR OBJ-7
STARTING GT STACKUP
GOAL G-1 PUTON1 BLOCK-3 ONTO TABLE-1
. GOAL G-2 CLEAROFF BLOCK-3
. . GOAL G-3 GETRIDOF PYRAMID-3
    REJECTING (426 17 0)
    LOOKING AT (400 17 0)
    FOUND REGION (300 0 0) TO (400 600 0)
. . . . GOAL G-4 PUT PYRAMID-3 (300 9 0)
. . . . . GOAL G-5 GRASP PYRAMID-3
. . . . . . GOAL G-6 GETRIDOF BLOCK-A
            LOOKING AT (965 340 0)
            FOUND REGION (600 200 0) TO (1200 800 0)
. . . . . . . GOAL G-7 PUT BLOCK-A (929 224 0)
. . . . . . . . GOAL G-8 GRASP BLOCK-A
                  G-8 SUCCEEDS
                (1) LIFTING BLOCK-A FROM (1000 575 101) TO (929 224 0)
                (2) LETTING GO OF BLOCK-A
                ADDING BLOCK-A ON TABLE-1 (POS)
                G-7 SUCCEEDS
              G-6 SUCCEEDS
. . . . . . GOAL G-9 CLEAROFF PYRAMID-3
```

```
            G-9 SUCCEEDS
          (3) MOVING HAND FROM (1029 349 100) TO (700 750 541)
          (4) GRASPING PYRAMID-3
          G-5 SUCCEEDS
        (5) LIFTING PYRAMID-3 FROM (650 700 301) TO (300 9 0)
        TAKING PYRAMID-3 FROM STACK-9
        STACK-9 DISMANTLED
        (6) LETTING GO OF PYRAMID-3
        ADDING PYRAMID-3 ON TABLE-1 (POS)
        G-4 SUCCEEDS
      G-3 SUCCEEDS
    G-2 SUCCEEDS
  REJECTING (41 141 0)
  LOOKING AT (41 157 0)
  REGION AT (0 157 0) TOO SMALL
  LOOKING AT (350 844 0)
  FOUND REGION (300 474 0) TO (600 1200 0)
. GOAL G-10 PUT BLOCK-3 (306 670 0)
. . GOAL G-11 GRASP BLOCK-3
. . . GOAL G-12 CLEAROFF BLOCK-3
        G-12 SUCCEEDS
      (7) MOVING HAND FROM (350 59 240) TO (700 750 301)
      (8) GRASPING BLOCK-3
      G-11 SUCCEEDS
    (9) LIFTING BLOCK-3 FROM (600 600 1) TO (306 670 0)
    (10) LETTING GO OF BLOCK-3
    ADDING BLOCK-3 ON TABLE-1 (POS)
    G-10 SUCCEEDS
  G-1 SUCCEEDS
GOAL G-13 PUTON1 BLOCK-A ONTO BLOCK-3
. GOAL G-14 CLEAROFF BLOCK-A
    G-14 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-3
. GOAL G-15 PUT BLOCK-A (306 695 300)
. . GOAL G-16 GRASP BLOCK-A
. . . GOAL G-17 CLEAROFF BLOCK-A
        G-17 SUCCEEDS
      (12) MOVING HAND FROM (406 820 300) TO (1029 349 100)
      (13) GRASPING BLOCK-A
      G-16 SUCCEEDS
    (14) LIFTING BLOCK-A FROM (929 224 0) TO (306 695 300)
    (15) LETTING GO OF BLOCK-A
    ADDING BLOCK-A ON BLOCK-3 (POS)
    MAKING STACK STACK-12 BLOCK-A BLOCK-3
    G-15 SUCCEEDS
  G-13 SUCCEEDS
GOAL G-18 PUTON1 BLOCK-4 ONTO BLOCK-A
. GOAL G-19 CLEAROFF BLOCK-4
    G-19 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-A
. GOAL G-20 PUT BLOCK-4 (306 720 400)
. . GOAL G-21 GRASP BLOCK-4
. . . GOAL G-22 CLEAROFF BLOCK-4
        G-22 SUCCEEDS
      (17) MOVING HAND FROM (406 820 400) TO (900 700 401)
      (18) GRASPING BLOCK-4
      G-21 SUCCEEDS
    (19) LIFTING BLOCK-4 FROM (800 600 201) TO (306 720 400)
    TAKING BLOCK-4 FROM STACK-10
    STACK-10 DISMANTLED
    (20) LETTING GO OF BLOCK-4
    ADDING BLOCK-4 ON BLOCK-A (POS)
    ADDING BLOCK-4 TO STACK-12
    G-20 SUCCEEDS
  G-18 SUCCEEDS
GOAL G-23 PUTON1 BLOCK-9 ONTO BLOCK-4
. GOAL G-24 CLEAROFF BLOCK-9
    G-24 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-4
. GOAL G-25 PUT BLOCK-9 (306 720 600)
. . GOAL G-26 GRASP BLOCK-9
. . . GOAL G-27 CLEAROFF BLOCK-9
        G-27 SUCCEEDS
      (22) MOVING HAND FROM (406 820 600) TO (1000 1100 201)
      (23) GRASPING BLOCK-9
      G-26 SUCCEEDS
    (24) LIFTING BLOCK-9 FROM (900 1000 1) TO (306 720 600)
    (25) LETTING GO OF BLOCK-9
    ADDING BLOCK-9 TO STACK-12
    ADDING BLOCK-9 ON BLOCK-4 (POS)
    G-25 SUCCEEDS
  G-23 SUCCEEDS
GOAL G-28 PUTON1 BLOCK-1 ONTO BLOCK-9
```

```
. GOAL G-29 CLEAROFF BLOCK-1
    G-29 SUCCE.DS
  FOUND REGION CLEARTOP BLOCK-9
. GOAL G-30 PUT BLOCK-1 (356 770 800)
. . GOAL G-31 GRASP BLOCK-1
. . . GOAL G-32 CLEAROFF BLOCK-1
        G-32 SUCCEEDS
      (27) MOVING HAND FROM (406 820 800) TO (1050 950 501)
      (28) GRASPING BLOCK-1
      G-31 SUCCEEDS
    (29) LIFTING BLOCK-1 FROM (1000 900 401) TO (356 770 800)
    TAKING BLOCK-1 FROM STACK-11
    STACK-11 DISMANTLED
    (30) LETTING GO OF BLOCK-1
    ADDING BLOCK-1 ON BLOCK-9 (POS)
    ADDING BLOCK-1 TO STACK-12
    G-30 SUCCEEDS
  G-28 SUCCEEDS
GOAL G-33 PUTON1 PYRAMID-1 ONTO BLOCK-1
. GOAL G-34 CLEAROFF PYRAMID-1
    G-34 SUCCEEDS
  FOUND REGION CLEARTOP BLOCK-1
. GOAL G-35 PUT PYRAMID-1 (356 770 900)
. . GOAL G-36 GRASP PYRAMID-1
. . . GOAL G-37 CLEAROFF PYRAMID-1
        G-37 SUCCEEDS
      (32) MOVING HAND FROM (406 820 900) TO (50 107 100)
      (33) GRASPING PYRAMID-1
      G-36 SUCCEEDS
    (34) LIFTING PYRAMID-1 FROM (0 57 0) TO (356 770 900)
    (35) LETTING GO OF PYRAMID-1
    ADDING PYRAMID-1 TO STACK-12
    ADDING PYRAMID-1 ON BLOCK-1 (POS)
    G-35 SUCCEEDS
  G-33 SUCCEEDS
CANT COMPLETE STACK PYRAMID-3 . . .
GT SUCCEEDS

REPLY (1 (LEFT OUT PYRAMID-3))
```

```
...........................................................................
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
.                                LBP2              LBB6                   .
.                LBB9                                                     .
.                LGB4SGP1++H1                                             .
.                L+BASPB1                                                 .
.                LRB3                                                     .
.                                ++X1       LGB2          LBB8            .
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
.    LBB6        SBP3  LBB7                                               .
...........................................................................
```

```
(DUMP)
AVPESTP (OBJ-1 L4-1 SIZE LARGE POS) (OBJ-1 R5-1 COLOR RED POS)
  (OBJ-2 S9-1 SIZE SMALL POS) (OBJ-4 S15-1 SIZE SMALL POS)
  [illegible] B19-1 COLOR BLACK POS) (OBJ-6 L23-1 SIZE LARGE POS)
  (OBJ-6 G24-1 COLOR GREEN POS) (OBJ-7 S28-1 SIZE SMALL POS)
CHECKSTACKUP2 (BLOCK-3) (BLOCK-4) (BLOCK-9) (BLOCK-A) (PYRAMID-1) (PYRAMID-3)
CHOICECOUNT (8)
CHOICETIME (1 1) (2 1) (3 7) (4 12) (5 17) (6 22) (7 27) (8 32)
CLEARTOP (BLOCK-2) (BLOCK-5) (BLOCK-6) (BLOCK-7) (BLOCK-8) (PYRAMID-1)
  (PYRAMID-2) (PYRAMID-3)
CUROBJ (OBJ-7 MAIN)
CUROBJP (OBJ-7 MAIN)
DEFDET (A14-1) (A18-1) (A22-1) (A27-1) (A3-1) (A8-1)
DETSEEN (A14-1) (A18-1) (A22-1) (A27-1) (A3-1) (A8-1)
ENDMARK (LE-1) (RE-1)
EDIT (I12-1)
ERRREF (OBJ-1 B6-1) (OBJ-2 B10-1) (OBJ-3 I12-1) (OBJ-4 P16-1) (OBJ-5 B20-1)
  (OBJ-6 B25-1) (OBJ-7 P29-1)
EVENTTIME (36)
GETRIDCHOICE (1 G-4 1 TABLE-1 PYRAMID-3 300 9 0)
  (2 G-7 1 TABLE-1 BLOCK-A 929 224 0)
GS1 (S-1)
GTYPED (S-1)
HASAV (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS) (BLOCK-2 COLOR GREEN POS)
  (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS) (BLOCK-3 SIZE LARGE POS)
  (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS) (BLOCK-5 COLOR BLUE POS)
  (BLOCK-5 SIZE LARGE POS) (BLOCK-6 COLOR BLACK POS) (BLOCK-6 SIZE LARGE POS)
  (BLOCK-7 COLOR BLACK POS) (BLOCK-7 SIZE LARGE POS) (BLOCK-8 COLOR BLACK POS)
  (BLOCK-8 SIZE LARGE POS) (BLOCK-9 COLOR BLACK POS) (BLOCK-9 SIZE LARGE POS)
  (BLOCK-A SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
  (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
  (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
  (PYRAMID-3 SIZE SMALL POS)
HASLEVEL (G-1 1) (G-10 2) (G-11 3) (G-12 4) (G-13 1) (G-14 2) (G-15 2) (G-16 3)
  (G-17 4) (G-18 1) (G-19 2) (G-2 2) (G-20 2) (G-21 3) (G-22 4) (G-23 1)
  (G-24 2) (G-25 2) (G-26 3) (G-27 4) (G-28 1) (G-29 2) (G-3 3) (G-30 2)
  (G-31 3) (G-32 4) (G-33 1) (G-34 2) (G-35 2) (G-36 3) (G-37 4) (G-4 4) (G-5 5)
  (G-6 6) (G-7 7) (G-8 8) (G-9 6)
HASREL (BLOCK-1 ON BLOCK-9 POS) (BLOCK-2 IN BOX-1 POS) (BLOCK-3 ON TABLE-1 POS)
  (BLOCK-4 ON BLOCK-A POS) (BLOCK-5 IN BOX-1 POS) (BLOCK-6 ON TABLE-1 POS)
  (BLOCK-7 ON TABLE-1 POS) (BLOCK-8 IN BOX-1 POS) (BLOCK-9 ON BLOCK-4 POS)
  (BLOCK-A ON BLOCK-3 POS) (BOX-1 ON TABLE-1 POS) (PYRAMID-1 ON BLOCK-1 POS)
  (PYRAMID-2 IN BOX-1 POS) (PYRAMID-3 ON TABLE-1 POS)
HASSIZE (BLOCK-1 100 100 100) (BLOCK-2 200 200 200) (BLOCK-3 200 300 300)
  (BLOCK-4 200 200 200) (BLOCK-5 300 100 400) (BLOCK-6 200 200 200)
  (BLOCK-7 200 200 200) (BLOCK-8 200 200 200) (BLOCK-9 200 200 200)
  (BLOCK-A 200 250 100) (BOX-1 600 600 1) (PYRAMID-1 100 100 100)
  (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 240) (TABLE-1 1200 1200 0)
HASSUPERGOAL (G-10 G-1) (G-15 G-13) (G-20 G-18) (G-25 G-23) (G-30 G-28)
  (G-35 G-33) (G-4 G-3) (G-7 G-6)
IMPCHOICE (BLOCK-4) (BLOCK-9) (PYRAMID-1) (PYRAMID-3)
IMPINDEF (OBJ-1) (OBJ-2) (OBJ-4) (OBJ-5) (OBJ-6) (OBJ-7)
IMPOBJ (S-1 BLOCK-3) (S-1 BLOCK-1) (S-1 BLOCK-A) (S-1 PYRAMID-3) (S-1 BLOCK-9)
  (S-1 BLOCK-4) (S-1 PYRAMID-1)
IMPREL (S-1 ON ?)
IMPTYPE (S-1 STACK)
INSET (BLOCK-1 S-2) (BLOCK-3 S-2) (BLOCK-4 S-2) (BLOCK-9 S-2) (BLOCK-A S-2)
  (PYRAMID-1 S-2) (PYRAMID-3 S-2) (TABLE-1 S-2)
INSTACK (BLOCK-1 STACK-12) (BLOCK-3 STACK-12) (BLOCK-4 STACK-12)
  (BLOCK-9 STACK-12) (BLOCK-A STACK-12) (PYRAMID-1 STACK-12)
ISA (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK) (BLOCK-4 BLOCK)
  (BLOCK-5 BLOCK) (BLOCK-6 BLOCK) (BLOCK-7 BLOCK) (BLOCK-8 BLOCK)
  (BLOCK-9 BLOCK) (BLOCK-A BLOCK) (BOX-1 BOX) (HAND-1 HAND) (PYRAMID-1 PYRAMID)
  (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
ISAV (B19-1 COLOR BLACK POS) (G24-1 COLOR GREEN POS) (L23-1 SIZE LARGE POS)
  (L4-1 SIZE LARGE POS) (R5-1 COLOR RED POS) (S15-1 SIZE SMALL POS)
  (S28-1 SIZE SMALL POS) (S9-1 SIZE SMALL POS)
ISDEF (OBJ-1) (OBJ-2) (OBJ-4) (OBJ-5) (OBJ-6) (OBJ-7)
ISIMPER (A11-1) (A13-1) (A17-1) (A21-1) (A26-1) (A7-1) (S1-1) (U2-1)
ISNOUN (B10-1 BLOCK) (B20-1 BLOCK) (B25-1 BLOCK) (B6-1 BLOCK) (I12-1 IT)
  (P16-1 PYRAMID) (P29-1 PYRAMID)
LEFTOF (A11-1 I12-1) (A13-1 A14-1) (A14-1 S15-1) (A17-1 A18-1) (A18-1 B19-1)
  (A21-1 A22-1) (A22-1 L23-1) (A26-1 A27-1) (A27-1 S28-1) (A3-1 L4-1)
  (A7-1 A8-1) (A8-1 S9-1) (B10-1 A11-1) (B19-1 B20-1) (B20-1 A21-1)
  (B25-1 A26-1) (B6-1 A7-1) (G24-1 B25-1) (I12-1 A13-1) (L23-1 G24-1)
  (L4-1 R5-1) (LE-1 S1-1) (P16-1 A17-1) (P29-1 RE-1) (R5-1 B6-1) (S1-1 U2-1)
  (S15-1 P16-1) (S28-1 P29-1) (S9-1 B10-1) (U2-1 A3-1)
LOCAT (BLOCK-1 356 770 800) (BLOCK-2 800 600 1) (BLOCK-3 306 670 0)
  (BLOCK-4 306 720 400) (BLOCK-5 900 900 1) (BLOCK-6 100 0 0) (BLOCK-7 400 0 0)
  (BLOCK-8 1000 600 1) (BLOCK-9 306 720 600) (BLOCK-A 306 695 300)
  (BOX-1 600 600 0) (HAND-1 406 820 1000) (PYRAMID-1 356 770 900)
  (PYRAMID-2 600 900 1) (PYRAMID-3 300 9 0) (TABLE-1 0 0 0)
NEXT (G-1 (STACKUPSET GT S-2)) (G-11 (PUTMOVE G-10 BLOCK-3 306 670 0))
  (G-12 (GRASP1 G-11 BLOCK-3 700 750 301)) (G-13 (STACKUPSET GT S-2))
  (G-14 (FINDSPACE BLOCK-3 BLOCK-A 200 250 100))
  (G-16 (PUTMOVE G-15 BLOCK-A 306 695 300))
  (G-17 (GRASP1 G-16 BLOCK-A 1029 349 100)) (G-18 (STACKUPSET GT S-2))
  (G-19 (FINDSPACE BLOCK-A BLOCK-4 200 200 200))
  (G-2 (FINDSPACE TABLE-1 BLOCK-3 200 300 300))
  (G-21 (PUTMOVE G-20 BLOCK-4 306 720 400))
  (G-22 (GRASP1 G-21 BLOCK-4 500 700 401)) (G-23 (STACKUPSET GT S-2))
  (G-24 (FINDSPACE BLOCK-4 BLOCK-9 200 200 200))
  (G-26 (PUTMOVE G-25 BLOCK-9 306 720 600))
  (G-27 (GRASP1 G-26 BLOCK-9 1000 1100 201)) (G-28 (STACKUPSET GT S-2))
  (G-29 (FINDSPACE BLOCK-9 BLOCK-1 100 100 100)) (G-3 (CLEAROFF G-2 BLOCK-3))
  (G-31 (PUTMOVE G-30 BLOCK-1 356 770 800))
  (G-32 (GRASP1 G-31 BLOCK-1 1050 950 501)) (G-33 (STACKUPSET GT S-2))
  (G-34 (FINDSPACE BLOCK-1 PYRAMID-1 100 100 100))
  (G-36 (PUTMOVE G-35 PYRAMID-1 356 770 900))
  (G-37 (GRASP1 G-36 PYRAMID-1 50 107 100))
  (G-5 (PUTMOVE G-4 PYRAMID-3 300 9 0)) (G-6 (GRASP G-5 PYRAMID-3))
  (G-8 (PUTMOVE G-7 BLOCK-A 929 224 0)) (G-9 (GRASP1 G-5 PYRAMID-3 700 750 541)))
NEXTF (G-1 (FAILPUTONSTACK GT BLOCK-3 TABLE-1 S-2))
  (G-13 (FAILPUTONSTACK GT BLOCK-A BLOCK-3 S-2))
  (G-18 (FAILPUTONSTACK GT BLOCK-4 BLOCK-A S-2))
  (G-23 (FAILPUTONSTACK GT BLOCK-9 BLOCK-4 S-2))
  (G-28 (FAILPUTONSTACK GT BLOCK-1 BLOCK-9 S-2))
  (G-33 (FAILPUTONSTACK GT PYPAMID-1 BLOCK-1 S-2))
NPBOUND (RE-1)
NPGCHK (A14-1) (A18-1) (A22-1) (A27-1) (A3-1) (A8-1) (I12-1)
NPGCHK3 (A14-1) (A18-1) (A22-1) (A27-1) (A3-1) (A8-1) (I12-1)
NREPLY (1)
NPESTR (OBJ-1 B6-1 BLOCK) (OBJ-5 B20-1 BLOCK) (OBJ-6 B25-1 BLOCK)
OLDAV (B19-1) (G24-1) (L23-1) (L4-1) (R5-1) (S15-1) (S20-1) (S9-1)
PUTON1CHOICE (3 G-1 1 BLOCK-3 TABLE-1 306 670 0)
  (4 G-13 1 BLOCK-A BLOCK-3 306 695 300) (5 G-18 1 BLOCK-4 BLOCK-A 306 720 400)
  (6 G-23 1 BLOCK-9 BLOCK-4 306 720 600) (7 G-28 1 BLOCK-1 BLOCK-9 356 770 800)
  (8 G-33 1 PYPAMID-1 BLOCK-1 356 770 900)
REFERS (OBJ-1 BLOCK-3) (OBJ-2 BLOCK-1) (OBJ-3 BLOCK-A) (OBJ-4 PYRAMID-3)
  (OBJ-5 BLOCK-9) (OBJ-6 BLOCK-4) (OBJ-7 PYRAMID-1)
REPLY (1 (LEFT OUT PYRAMID-3))
SENTENCE (S-1)
TEXT
  (19
    (STACK UP A LARGE RED BLOCK AND A SMALL BLOCK AND IT AND A SMALL PYRAMID AND
        A BLACK BLOCK AND A LARGE GREEN BLOCK AND A SMALL PYRAMID))
TPACING (T)
TRIEDSTACK (BLOCK-1 S-2) (BLOCK-3 S-2) (BLOCK-4 S-2) (BLOCK-9 S-2) (BLOCK-A S-2)
  (PYPAMID-1 S-2) (TABLE-1 S-2)
UNEVENT (0 (EPSTPIEDSTACK BLOCK-3 S-2)) (1 (MOVEHAND 1100 700 201))
  (2 (GRASP3 HAND-1 BLOCK-A)) (3 (MOVEHAND 1029 349 100))
  (4 (UNGRASP PYPAMID-3)) (5 (MOVEHAND 700 750 541))
  (6 (GRASP3 HAND-1 PYPAMID-3)) (7 (MOVEHAND 350 59 240)) (8 (UNGRASP BLOCK-3))
  (9 (MOVEHAND 700 750 301)) (10 (GRASP3 HAND-1 BLOCK-3))
  (11 (EPSTRIEDSTACK BLOCK-A S-2)) (12 (MOVEHAND 406 820 300))
  (13 (UNGRASP BLOCK-A)) (14 (MOVEHAND 1029 349 100))
  (15 (GRASP3 HAND-1 BLOCK-A)) (16 (EPSTPIEDSTACK BLOCK-4 S-2))
  (17 (MOVEHAND 406 820 400)) (18 (UNGPASP BLOCK-4)) (19 (MOVEHAND 900 700 401))
  (20 (GPASP3 HAND-1 BLOCK-4)) (21 (EPSTPIEDSTACK BLOCK-9 S-2))
  (22 (MOVEHAND 406 820 600)) (23 (UNGPASP BLOCK-9))
  (24 (MOVEHAND 1000 1100 201)) (25 (GRASP3 HAND-1 BLOCK-9))
  (26 (EPSTPIEDSTACK BLOCK-1 S-2)) (27 (MOVEHAND 406 820 800))
  (28 (UNGRASP BLOCK-1)) (29 (MOVEHAND 1050 950 501))
  (30 (GPASP3 HAND-1 BLOCK-1)) (31 (EPSTPIEDSTACK PYPAMID-1 S-2))
  (32 (MOVEHAND 406 820 900)) (33 (UNGPASP PYPAMID-1))
  (34 (MOVEHAND 50 107 100)) (35 (GPASP3 HAND-1 PYPAMID-1)))
WORDEQ (A11-1 AND) (A13-1 AND) (A14-1 A) (A17-1 AND) (A18-1 A) (A21-1 AND)
  (A22-1 A) (A26-1 AND) (A27-1 A) (A3-1 A) (A7-1 AND) (A8-1 A) (B10-1 BLOCK)
  (B19-1 BLACK) (B20-1 BLOCK) (B25-1 BLOCK) (B6-1 BLOCK) (G24-1 GREEN)
  (I12-1 IT) (L23-1 LARGE) (L4-1 LARGE) (P16-1 PYRAMID) (P29-1 PYRAMID)
  (R5-1 RED) (S1-1 STACK) (S15-1 SMALL) (S20-1 SMALL) (S9-1 SMALL) (U2-1 UP)
ok
```

```
.........................................................................
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                           LBP2              LBB5                      .
.             LBB9                                                      .
.             LGB4SGP1++H1                                              .
.             L+BASRB1                                                  .
.             LRB3                                                      .
.                           ++X1       LGB2           LBB8              .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.     LBB6          SRP3  LBB7                                          .
.........................................................................
```

```
ADDING SIZE LAPGE (POS) TO BLOCK-0
ADDING BLOCK BLOCK-0
20 INPUT TEXT IS " PUT IT DOWN "
OBJ-1 REFERS BLOCK-0
STARTING GT PUT BLOCK-0 DOWN
GOAL G-1 GETRIDOF BLOCK-0
```

```
LOOKING AT (80 750 0)
FOUND REGION (0 200 0) TO (306 800 0)
. GOAL G-2 PUT BLOCK-0 (2 254 0)
. . GOAL G-3 GRASP BLOCK-0
      G-3 SUCCEEDS
    (0) LIFTING BLOCK-0 FROM (256 670 900) TO (2 254 0)
    ADDING BLOCK-0 ON TABLE-1 (POS)
    (1) LETTING GO OF BLOCK-0
    G-2 SUCCEEDS
  G-1 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))
```

```
.........................................................................
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                     LBP2              LBB5            .
.                 LBB9                                                  .
.                 LGB4SGP1                                              .
.                 L+BASRB1                                              .
.                 LRB3                                                  .
.                                     ++X1       LGB2         LBB8      .
.                                                                       .
.                                                                       .
.                                                                       .
.        ++H1                                                           .
.                                                                       .
L+B0                                                                    .
.                                                                       .
.                                                                       .
.                                                                       .
.    LBB6        SRP3  LBB7                                             .
.........................................................................
```

```
21 INPUT TEXT IS " PUT THE LARGE BLUE BLOCK AND THE LARGE PYRAMID ON THE TABLE
  "
OBJ-1 AMBIG L3-1 BLOCK-0 BLOCK-2 ...
OBJ-1 AMBIG B4-1 BLOCK-5 PYRAMID-2 ...
OBJ-1 REFEPS BLOCK-5
OBJ-2 AMBIG L8-1 BLOCK-0 BLOCK-2 ...
OBJ-2 REFEPS PYPAMID-2
OBJ-3 REFEPS TABLE-1
RELINCON OBJ-2 P9-1 ON TABLE-1 POS
DOING GT PUTON SET S-2 (TABLE-1)
GOAL G-1 PUTON PYPAMID-2 ONTO TABLE-1
. GOAL G-2 CLEAROFF PYRAMID-2
    G-2 SUCCEEDS
  REJECTING (205 204 0)
  LOOKING AT (205 254 0)
  REGION AT (0 200 0) TOO SMALL
  LOOKING AT (334 153 0)
  REGION AT (302 109 0) TOO SMALL
  LOOKING AT (930 250 0)
  FOUND REGION (600 200 0) TO (1200 600 0)
. GOAL G-3 PUT PYPAMID-2 (800 229 0)
. . GOAL G-4 GRASP PYPAMID-2
. . . GOAL G-5 CLEAROFF PYPAMID-2
        G-5 SUCCEEDS
      (1) MOVING HAND FROM (152 404 100) TO (750 1000 201)
      (2) GRASPING PYRAMID-2
      G-4 SUCCEEDS
    (3) LIFTING PYRAMID-2 FROM (600 900 1) TO (800 229 0)
    (4) LETTING GO OF PYRAMID-2
    ADDING PYPAMID-2 ON TABLE-1 (POS)
    G-3 SUCCEEDS
  G-1 SUCCEEDS
DOING GT PUTON SET S-2 (TABLE-1)
GOAL G-6 PUTON BLOCK-5 ONTO TABLE-1
. GOAL G-7 CLEAROFF BLOCK-5
    G-7 SUCCEEDS
  LOOKING AT (512 203 0)
  FOUND PEGION (506 200 0) TO (800 800 0)
. GOAL G-8 PUT BLOCK-5 (545 216 0)
. . GOAL G-9 GRASP BLOCK-5
. . . GOAL G-10 CLEAROFF BLOCK-5
```

```
      G-10 SUCCEEDS
     (6) MOVING HAND FROM (1030 329 200) TO (1050 950 401))
     (7) GRASPING BLOCK-5
    G-9 SUCCEEDS
   (8) LIFTING BLOCK-5 FROM (900 900 1) TO (545 216 0)
   (9) LETTING GO OF BLOCK-5
   ADDING BLOCK-5 ON TABLE-1 (POS)
   G-8 SUCCEEDS
  G-6 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))



CLEARTOP (BLOCK-0) (BLOCK-2) (BLOCK-5) (BLOCK-6) (BLOCK-7) (BLOCK-8) (PYRAMID-1)
  (PYRAMID-2) (PYRAMID-3)
HASAV (BLOCK-0 SIZE LARGE POS) (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS)
  (BLOCK-2 COLOR GREEN POS) (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS)
  (BLOCK-3 SIZE LARGE POS) (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS)
  (BLOCK-5 COLOR BLUE POS) (BLOCK-5 SIZE LARGE POS) (BLOCK-6 COLOR BLACK POS)
  (BLOCK-6 SIZE LARGE POS) (BLOCK-7 COLOR BLACK POS) (BLOCK-7 SIZE LARGE POS)
  (BLOCK-8 COLOR BLACK POS) (BLOCK-8 SIZE LARGE POS) (BLOCK-9 COLOR BLACK POS)
  (BLOCK-9 SIZE LARGE POS) (BLOCK-A SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
  (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
  (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
  (PYRAMID-3 SIZE SMALL POS)
HASREL (BLOCK-0 ON TABLE-1 POS) (BLOCK-1 ON BLOCK-9 POS) (BLOCK-2 IN BOX-1 POS)
  (BLOCK-3 ON TABLE-1 POS) (BLOCK-4 ON BLOCK-A POS) (BLOCK-5 ON TABLE-1 POS)
  (BLOCK-6 ON TABLE-1 POS) (BLOCK-7 ON TABLE-1 POS) (BLOCK-8 IN BOX-1 POS)
  (BLOCK-9 ON BLOCK-4 POS) (BLOCK-A ON BLOCK-3 POS) (BOX-1 ON TABLE-1 POS)
  (PYRAMID-1 ON BLOCK-1 POS) (PYRAMID-2 ON TABLE-1 POS)
  (PYRAMID-3 ON TABLE-1 POS)
HASSIZE (BLOCK-0 300 300 100) (BLOCK-1 100 100 100) (BLOCK-2 200 200 200)
  (BLOCK-3 200 300 300) (BLOCK-4 200 200 200) (BLOCK-5 300 100 400)
  (BLOCK-6 200 200 200) (BLOCK-7 200 200 200) (BLOCK-8 200 200 200)
  (BLOCK-9 200 200 200) (BLOCK-A 200 250 100) (BOX-1 600 600 1)
  (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 240)
  (TABLE-1 1200 1200 0)
INSTACK (BLOCK-1 STACK-12) (BLOCK-3 STACK-12) (BLOCK-4 STACK-12)
  (BLOCK-9 STACK-12) (BLOCK-A STACK-12) (PYRAMID-1 STACK-12)
ISA (BLOCK-0 BLOCK) (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK)
  (BLOCK-4 BLOCK) (BLOCK-5 BLOCK) (BLOCK-6 BLOCK) (BLOCK-7 BLOCK)
  (BLOCK-8 BLOCK) (BLOCK-9 BLOCK) (BLOCK-A BLOCK) (BOX-1 BOX) (HAND-1 HAND)
  (PYRAMID-1 PYRAMID) (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-0 2 254 0) (BLOCK-1 356 770 800) (BLOCK-2 800 600 1)
  (BLOCK-3 306 670 0) (BLOCK-4 306 720 400) (BLOCK-5 545 216 0)
  (BLOCK-6 100 0 0) (BLOCK-7 400 0 0) (BLOCK-8 1000 600 1) (BLOCK-9 306 720 600)
  (BLOCK-A 306 695 300) (BOX-1 600 600 0) (HAND-1 695 266 400)
  (PYRAMID-1 356 770 900) (PYRAMID-2 800 229 0) (PYRAMID-3 300 9 0)
  (TABLE-1 0 0 0)

.........................................................................
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                 LB09                                                  .
.                 LGB4SGP1                                              .
.                 L+BASPB1                                              .
.                 LRB3                                                  .
.                                   ++X1        LGB2        LBB8        .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
L+B0                                      ++H1                          .
.                                 LBB5                 LBP2             .
.                                                                       .
.                                                                       .
.     LBB6            SRP3  LBB7                                        .
.........................................................................



RUN TIME 14 MIN. 50.6 SEC

EXAM    TRY     FIRE    WMACT   E/F    E/T    T/F
5022    1300    933     3124    6.24   4.45   1.40
0.153   0.601   0.955   0.205   SEC AVG
```

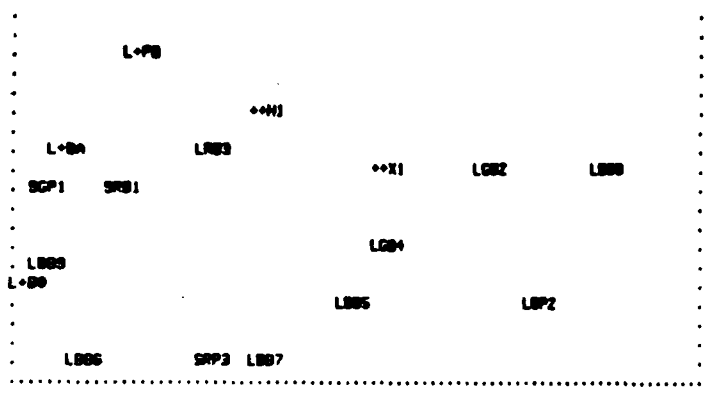
```
1912 INSERTS 1212 DELETES 576 WARNINGS 23 NEW OBJECTS
MAX :SMPX LENGTH 214
CORE (FREE.FULL): (12094 . 2508) USED (3204 . 462)
FIRED 86 OUT OF 400 PRODS


                    - - - - - - - - - - - - - - - - - - - -


EIGHTH SEGMENT

ADDING SIZE LARGE (POS) TO PYRAMID-B
ADDING PYRAMID PYRAMID-B
22 INPUT TEXT IS " PUT IT DOWN "
OBJ-1 REFERS PYRAMID-B
STARTING GT PUT PYRAMID-B DOWN
GOAL G-1 GETRIDOF PYRAMID-B
REJECTING (894 799 0)
LOOKING AT (894 600 0)
REGION AT (845 554 0) TOO SMALL
REJECTING (857 821 0)
LOOKING AT (857 600 0)
REGION AT (845 554 0) TOO SMALL
LOOKING AT (94 162 0)
REGION AT (0 109 0) TOO SMALL
LOOKING AT (720 475 0)
REGION AT (600 429 0) TOO SMALL
LOOKING AT (29 958 0)
REGION AT (0 554 0) TOO SMALL
REJECTING (187 1 0)
LOOKING AT (100 1 0)
REGION AT (0 0 0) TOO SMALL
REJECTING (139 94 0)
LOOKING AT (100 94 0)
REGION AT (0 0 0) TOO SMALL
REJECTING (686 665 0)
LOOKING AT (686 600 0)
REGION AT (600 554 0) TOO SMALL
REJECTING (56 330 0)
LOOKING AT (2 330 0)
REGION AT (0 316 0) TOO SMALL
REJECTING (106 17 0)
LOOKING AT (100 17 0)
REGION AT (0 0 0) TOO SMALL
LOOKING AT (103 799 0)
REGION AT (0 554 0) TOO SMALL
LOOKING AT (117 584 0)
REGION AT (0 554 0) TOO SMALL
LOOKING AT (260 660 0)
REGION AT (0 554 0) TOO SMALL
LOOKING AT (128 1053 0)
FOUND REGION (0 970 0) TO (600 1200 0)
. GOAL G-2 PUT PYRAMID-B (183 974 0)
. . GOAL G-3 GRASP PYRAMID-B
      G-3 SUCCEEDS
    (0) LIFTING PYRAMID-B FROM (456 156 300) TO (183 974 0)
    (1) LETTING GO OF PYRAMID-B
    ADDING PYRAMID-B ON TABLE-1 (POS)
    G-2 SUCCEEDS
  G-1 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))


.........................................................................
.                                                                       
.                         ++H1                                          
.                                                                       
.          L+PB                                                         
.                                                                       
.                 LB09                                                  
.                 LGB4SGP1                                              
.                 L+BASPB1                                              
.                 LRB3                                                  .
.                                   ++X1        LGB2        LBB0        .
.                                                                       .
.                                                                       .
.                                                                       .
.                                                                       .
L+B0                                                                    .
.                                 LBB5                 LBP2             .
.                                                                  .
.                                                                  .
.     L006          SRP3  L007                                     .
....................................................................


24 INPUT TEXT IS " PICK UP THE LARGE RED BLOCK "
OBJ-1 AMBIG L4-1 BLOCK-0 BLOCK-2 ...
OBJ-1 REFERS BLOCK-3
STARTING GT PICKUP BLOCK-3
GOAL G-1 GRASP BLOCK-3
. GOAL G-2 CLEAROFF BLOCK-3
. . GOAL G-3 GETRIDOF BLOCK-A
    REJECTING (965 869 0)
    LOOKING AT (965 600 0)
    REGION AT (845 554 0) TOO SMALL
    REJECTING (267 117 0)
    LOOKING AT (300 117 0)
    REGION AT (300 109 0) TOO SMALL
    REJECTING (905 327 0)
    LOOKING AT (888 327 0)
    REGION AT (845 316 0) TOO SMALL
    LOOKING AT (972 581 0)
    REGION AT (845 554 0) TOO SMALL
    LOOKING AT (869 453 0)
    REGION AT (845 429 0) TOO SMALL
    LOOKING AT (268 625 0)
    FOUND REGION (0 554 0) TO (306 974 0)
. . . GOAL G-4 PUT BLOCK-A (66 600 0)
. . . . GOAL G-5 GRASP BLOCK-A
. . . . . GOAL G-6 CLEAROFF BLOCK-A
. . . . . . GOAL G-7 GETRIDOF BLOCK-4
            LOOKING AT (881 577 0)
            REGION AT (845 554 0) TOO SMALL
            REJECTING (909 1044 0)
            LOOKING AT (600 1044 0)
            REGION AT (600 970 0) TOO SMALL
            LOOKING AT (565 697 0)
            REGION AT (506 554 0) TOO SMALL
            LOOKING AT (735 325 0)
            FOUND REGION (600 316 0) TO (888 600 0)
. . . . . . . GOAL G-8 PUT BLOCK-4 (618 380 0)
. . . . . . . . GOAL G-9 GRASP BLOCK-4
. . . . . . . . . GOAL G-10 CLEAROFF BLOCK-4
. . . . . . . . . . GOAL G-11 GETRIDOF BLOCK-9
                    LOOKING AT (521 229 0)
                    REGION AT (506 200 0) TOO SMALL
                    REJECTING (117 80 0)
                    LOOKING AT (100 80 0)
                    REGION AT (0 0 0) TOO SMALL
                    REJECTING (963 400 0)
                    LOOKING AT (963 429 0)
                    REGION AT (845 429 0) TOO SMALL
                    LOOKING AT (645 122 0)
                    REGION AT (600 109 0) TOO SMALL
                    LOOKING AT (998 564 0)
                    REGION AT (845 554 0) TOO SMALL
                    REJECTING (965 741 0)
                    LOOKING AT (965 600 0)
                    REGION AT (845 554 0) TOO SMALL
                    REJECTING (409 957 0)
                    FINDSPACE LIMIT EXCEEDED
                    TRYING ON BLOCK-0
                    FOUND REGION CLEARTOP BLOCK-0
. . . . . . . . . . . GOAL G-12 PUT BLOCK-9 (52 304 100)
. . . . . . . . . . . . GOAL G-13 GRASP BLOCK-9
. . . . . . . . . . . . . GOAL G-14 CLEAROFF BLOCK-9
. . . . . . . . . . . . . . GOAL G-15 GETRIDOF BLOCK-1
                            LOOKING AT (857 389 0)
                            REGION AT (845 316 0) TOO SMALL
                            REJECTING (1068 693 0)
                            LOOKING AT (1068 600 0)
                            REGION AT (845 554 0) TOO SMALL
                            REJECTING (442 774 0)
                            LOOKING AT (506 774 0)
                            REGION AT (506 554 0) TOO SMALL
                            LOOKING AT (18 868 0)
                            FOUND REGION (0 554 0) TO (183 670 0)
. . . . . . . . . . . . . . . GOAL G-16 PUT BLOCK-1 (41 554 0)
. . . . . . . . . . . . . . . . GOAL G-17 GRASP BLOCK-1
. . . . . . . . . . . . . . . . . GOAL G-18 CLEAROFF BLOCK-1
. . . . . . . . . . . . . . . . . . GOAL G-19 GETRIDOF PYRAMID-1
                                    LOOKING AT (1045 533 0)
                                    FOUND REGION (845 429 0) TO (1200 600 0)
. . . . . . . . . . . . . . . . . . . GOAL G-20 PUT PYRAMID-1 (1024 449 0)
. . . . . . . . . . . . . . . . . . . . GOAL G-21 GRASP PYRAMID-1
. . . . . . . . . . . . . . . . . . . . . GOAL G-22 CLEAROFF PYRAMID-1
                                              G-22 SUCCEEDS
                                             (0) MOVING HAND FROM (303 1004 100)
 TO (406 820 1000)
                                             (1) GRASPING PYRAMID-1
                                            G-21 SUCCEEDS
                                          (2) LIFTING PYRAMID-1 FROM (356 770 900)
  TO (1024 449 0)
                                          TAKING PYRAMID-1 FROM STACK-12
                                          ADDING PYRAMID-1 ON TABLE-1 (POS)
                                          (3) LETTING GO OF PYRAMID-1
                                          G-20 SUCCEEDS
                                        G-19 SUCCEEDS
                                      G-18 SUCCEEDS
                                    (4) MOVING HAND FROM (1074 490 100) TO (406
 820 900)
                                    (5) GRASPING BLOCK-1
                                    G-17 SUCCEEDS
                                  (6) LIFTING BLOCK-1 FROM (356 770 800) TO (41
 554 0)
                                  TAKING BLOCK-1 FROM STACK-12
                                  (7) LETTING GO OF BLOCK-1
                                  ADDING BLOCK-1 ON TABLE-1 (POS)
                                  G-16 SUCCEEDS
                                G-15 SUCCEEDS
                              G-14 SUCCEEDS
                            (8) MOVING HAND FROM (91 604 100) TO (406 820 800)
                            (9) GRASPING BLOCK-9
                            G-13 SUCCEEDS
                          (10) LIFTING BLOCK-9 FROM (306 720 600) TO (52 304 100)
                          TAKING BLOCK-9 FROM STACK-12
                          (11) LETTING GO OF BLOCK-9
                          ADDING BLOCK-9 ON BLOCK-0 (POS)
                          MAKING STACK STACK-13 BLOCK-9 BLOCK-0
                          G-12 SUCCEEDS
                        G-11 SUCCEEDS
                      G-10 SUCCEEDS
                    (12) MOVING HAND FROM (152 404 300) TO (406 820 600)
                    (13) GRASPING BLOCK-4
                    G-9 SUCCEEDS
                  (14) LIFTING BLOCK-4 FROM (306 720 400) TO (618 380 0)
                  TAKING BLOCK-4 FROM STACK-12
                  (15) LETTING GO OF BLOCK-4
                  ADDING BLOCK-4 ON TABLE-1 (POS)
                  G-8 SUCCEEDS
                G-7 SUCCEEDS
              G-6 SUCCEEDS
            (16) MOVING HAND FROM (718 400 200) TO (406 820 400)
            (17) GRASPING BLOCK-A
            G-5 SUCCEEDS
          MOVE TO (166 733 100) OVERLAPS BLOCK-A WITH BLOCK-1
          (18) LETTING GO OF BLOCK-A
          G-4 SUCCEEDS
        G-3 SUCCEEDS
. . GOAL G-23 GETRIDOF BLOCK-A
    REJECTING (346 810 0)
    LOOKING AT (306 810 0)
    REGION AT (302 654 0) TOO SMALL
    REJECTING (732 749 0)
    LOOKING AT (600 749 0)
    REGION AT (600 654 0) TOO SMALL
    REJECTING (54 524 0)
    LOOKING AT (54 554 0)
    REGION AT (0 554 0) TOO SMALL
    LOOKING AT (207 893 0)
    REGION AT (141 654 0) TOO SMALL
    LOOKING AT (153 670 0)
    REGION AT (141 654 0) TOO SMALL
    REJECTING (597 18 0)
    LOOKING AT (600 18 0)
    REGION AT (600 0 0) TOO SMALL
    FINDSPACE LIMIT EXCEEDED
    TRYING ON BLOCK-0
    REJECTING (152 346 100)
    LOOKING AT (152 304 100)
    REGION AT (2 254 100) TOO SMALL
    REJECTING (54 329 100)
```

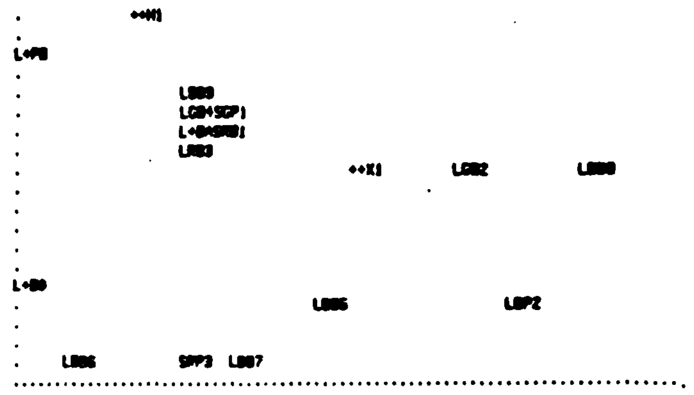
```
LOOKING AT (52 329 100)
REGION AT (2 254 100) TOO SMALL
LOOKING AT (50 263 100)
REGION AT (2 254 100) TOO SMALL
REJECTING (116 344 100)
LOOKING AT (116 304 100)
REGION AT (2 254 100) TOO SMALL
LOOKING AT (52 317 100)
REGION AT (2 254 100) TOO SMALL
LOOKING AT (87 302 100)
REGION AT (2 254 100) TOO SMALL
REJECTING (142 373 100)
FINDSPACE LIMIT EXCEEDED
G-23 EXHAUSTED
(18) GRASPING BLOCK-A
(17) LETTING GO OF BLOCK-A
(16) MOVING HAND FROM (406 820 400) TO (710 400 200)
(15) GRASPING BLOCK-4
(14) LIFTING BLOCK-4 FROM (610 300 0) TO (306 720 400)
ADDING BLOCK-4 ON BLOCK-A (POS)
ADDING BLOCK-4 TO STACK-12
(13) LETTING GO OF BLOCK-4
(12) MOVING HAND FROM (406 820 600) TO (152 404 300)
(11) GRASPING BLOCK-9
(10) LIFTING BLOCK-9 FROM (52 304 100) TO (306 720 600)
TAKING BLOCK-9 FROM STACK-13
STACK-13 DISMANTLED
ADDING BLOCK-9 TO STACK-12
ADDING BLOCK-9 ON BLOCK-4 (POS)
(9) LETTING GO OF BLOCK-9
(8) MOVING HAND FROM (406 820 800) TO (91 604 100)
(7) GRASPING BLOCK-1
(6) LIFTING BLOCK-1 FROM (41 554 0) TO (356 770 800)
ADDING BLOCK-1 ON BLOCK-9 (POS)
ADDING BLOCK-1 TO STACK-12
(5) LETTING GO OF BLOCK-1
(4) MOVING HAND FROM (406 820 900) TO (1074 459 100)
(3) GRASPING PYRAMID-1
(2) LIFTING PYRAMID-1 FROM (1024 449 0) TO (356 770 900)
ADDING PYRAMID-1 TO STACK-12
ADDING PYRAMID-1 ON BLOCK-1 (POS)
(1) LETTING GO OF PYRAMID-1
(0) MOVING HAND FROM (406 820 1000) TO (383 1084 100)
. . . . . . . . . . . . . . . . . . . . GOAL G-20 RETRY GETRIDOF PYRAMID-1
                                      LOOKING AT (687 55 0)
                                      FOUND REGION (600 0 0) TO (800 216 0)
. . . . . . . . . . . . . . . . . . . . . GOAL G-24 PUT PYRAMID-1 (670 63 0)
. . . . . . . . . . . . . . . . . . . . . . GOAL G-25 GRASP PYRAMID-1
. . . . . . . . . . . . . . . . . . . . . . . GOAL G-26 CLEAROFF PYRAMID-1
                                                G-26 SUCCEEDS
                                               (0) MOVING HAND FROM (383 1084 100)
 TO (406 820 1000)
                                               (1) GRASPING PYRAMID-1
                                              G-25 SUCCEEDS
                                             (2) LIFTING PYRAMID-1 FROM (356 770
900) TO (670 63 0)
                                             TAKING PYRAMID-1 FROM STACK-12
                                             (3) LETTING GO OF PYRAMID-1
                                             ADDING PYRAMID-1 ON TABLE-1 (POS)
                                             G-24 SUCCEEDS
                                            G-20 SUCCEEDS
                                           G-19 SUCCEEDS
                                          G-18 SUCCEEDS
                                         (4) MOVING HAND FROM (720 113 100) TO (406
820 900)
                                         (5) GRASPING BLOCK-1
                                         G-17 SUCCEEDS
                                        (6) LIFTING BLOCK-1 FROM (356 770 800) TO (41
554 0)
                                        TAKING BLOCK-1 FROM STACK-12
                                        (7) LETTING GO OF BLOCK-1
                                        ADDING BLOCK-1 ON TABLE-1 (POS)
                                        G-16 SUCCEEDS
                                       G-15 SUCCEEDS
                                      G-14 SUCCEEDS
                                     (8) MOVING HAND FROM (91 604 100) TO (406 820 800)
                                     (9) GRASPING BLOCK-9
                                     G-13 SUCCEEDS
                                    (10) LIFTING BLOCK-9 FROM (306 720 600) TO (52 304 100)
                                    TAKING BLOCK-9 FROM STACK-12
                                    (11) LETTING GO OF BLOCK-9
                                    ADDING BLOCK-9 ON BLOCK-0 (POS)
                                    MAKING STACK STACK-14 BLOCK-9 BLOCK-0
```

```
                G-12 SUCCEEDS
               G-11 SUCCEEDS
              G-10 SUCCEEDS
             (12) MOVING HAND FROM (152 404 300) TO (406 820 800)
             (13) GRASPING BLOCK-4
             G-9 SUCCEEDS
            (14) LIFTING BLOCK-4 FROM (306 720 400) TO (610 300 0)
            TAKING BLOCK-4 FROM STACK-12
            (15) LETTING GO OF BLOCK-4
            ADDING BLOCK-4 ON TABLE-1 (POS)
            G-8 SUCCEEDS
           G-7 SUCCEEDS
          G-6 SUCCEEDS
         (16) MOVING HAND FROM (710 400 200) TO (406 820 400)
         (17) GRASPING BLOCK-A
         G-5 SUCCEEDS
        MOVE TO (166 733 100) OVERLAPS BLOCK-A WITH BLOCK-1
        (18) LETTING GO OF BLOCK-A
        G-4 SUCCEEDS
       G-3 SUCCEEDS
. . . GOAL G-27 GETRIDOF BLOCK-A
      REJECTING (992 634 0)
      LOOKING AT (992 600 0)
      REGION AT (845 580 0) TOO SMALL
      REJECTING (309 777 0)
      LOOKING AT (306 777 0)
      REGION AT (302 654 0) TOO SMALL
      LOOKING AT (711 357 0)
      REGION AT (600 316 0) TOO SMALL
      REJECTING (265 522 0)
      LOOKING AT (265 554 0)
      REGION AT (141 554 0) TOO SMALL
      REJECTING (996 981 0)
      LOOKING AT (996 600 0)
      REGION AT (845 580 0) TOO SMALL
      LOOKING AT (867 206 0)
      REGION AT (845 200 0) TOO SMALL
      FINDSPACE LIMIT EXCEEDED
      TRYING ON BLOCK-0
      LOOKING AT (167 275 100)
      REGION AT (2 254 100) TOO SMALL
      REJECTING (101 324 100)
      LOOKING AT (101 304 100)
      REGION AT (2 254 100) TOO SMALL
      LOOKING AT (42 322 100)
      REGION AT (2 254 100) TOO SMALL
      LOOKING AT (140 269 100)
      REGION AT (2 254 100) TOO SMALL
      REJECTING (92 335 100)
      LOOKING AT (92 304 100)
      REGION AT (2 254 100) TOO SMALL
      REJECTING (140 351 100)
      LOOKING AT (140 304 100)
      REGION AT (2 254 100) TOO SMALL
      REJECTING (144 378 100)
      FINDSPACE LIMIT EXCEEDED
      G-27 EXHAUSTED
      (18) GRASPING BLOCK-A
      (17) LETTING GO OF BLOCK-A
      (16) MOVING HAND FROM (406 820 400) TO (710 400 200)
      (15) GRASPING BLOCK-4
      (14) LIFTING BLOCK-4 FROM (610 300 0) TO (306 720 400)
      ADDING BLOCK-4 ON BLOCK-A (POS)
      ADDING BLOCK-4 TO STACK-12
      (13) LETTING GO OF BLOCK-4
      (12) MOVING HAND FROM (406 820 600) TO (152 404 300)
      (11) GRASPING BLOCK-9
      (10) LIFTING BLOCK-9 FROM (52 304 100) TO (306 720 600)
      TAKING BLOCK-9 FROM STACK-14
      STACK-14 DISMANTLED
      ADDING BLOCK-9 TO STACK-12
      ADDING BLOCK-9 ON BLOCK-4 (POS)
      (9) LETTING GO OF BLOCK-9
      (8) MOVING HAND FROM (406 820 800) TO (91 604 100)
      (7) GRASPING BLOCK-1
      (6) LIFTING BLOCK-1 FROM (41 554 0) TO (356 770 800)
      ADDING BLOCK-1 ON BLOCK-9 (POS)
      ADDING BLOCK-1 TO STACK-12
      (5) LETTING GO OF BLOCK-1
      (4) MOVING HAND FROM (406 820 900) TO (720 113 100)
      (3) GRASPING PYRAMID-1
      (2) LIFTING PYRAMID-1 FROM (670 63 0) TO (356 770 900)
      ADDING PYRAMID-1 TO STACK-12
```

```
  ADDING PYRAMID-1 ON BLOCK-1 (POS)
  (1) LETTING GO OF PYRAMID-1
  (0) MOVING HAND FROM (406 820 1000) TO (303 1004 100)
. . . . . . . . . . . . . . . . . . . . . . GOAL G-20 RETRY GETRIDOF PYRAMID-1
                                            REJECTING (1005 1091 0)
                                            LOOKING AT (600 1091 0)
                                            REGION AT (600 970 0) TOO SMALL
                                            LOOKING AT (296 679 0)
                                            FOUND REGION (0 554 0) TO (183 670 0)
. . . . . . . . . . . . . . . . . . . . . . . GOAL G-28 PUT PYRAMID-1 (55 562 0)
. . . . . . . . . . . . . . . . . . . . . . . . GOAL G-29 GRASP PYRAMID-1
. . . . . . . . . . . . . . . . . . . . . . . . . GOAL G-30 CLEAROFF PYRAMID-1
                                                    G-30 SUCCEEDS
                                                  (0) MOVING HAND FROM (303 1004 100)
 TO (406 820 1000)

                                                  (1) GRASPING PYRAMID-1
                                                  G-29 SUCCEEDS
                                                (2) LIFTING PYRAMID-1 FROM (356 770
900) TO (55 562 0)

                                                TAKING PYRAMID-1 FROM STACK-12
                                                (3) LETTING GO OF PYRAMID-1
                                                ADDING PYRAMID-1 ON TABLE-1 (POS)
                                                G-28 SUCCEEDS
                                              G-29 SUCCEEDS
                                            G-19 SUCCEEDS
                                          G-18 SUCCEEDS
                                        (4) MOVING HAND FROM (105 612 100) TO (406
820 900)

                                        (5) GRASPING BLOCK-1
                                        G-17 SUCCEEDS
                                      MOVE TO (91 604 100) OVERLAPS BLOCK-1 WITH
PYRAMID-1

                                      (6) LETTING GO OF BLOCK-1
                                      G-16 SUCCEEDS
                                    G-15 SUCCEEDS
. . . . . . . . . . . . . . . . . . GOAL G-31 GETRIDOF BLOCK-1
                                  REJECTING (367 777 0)
                                  LOOKING AT (306 777 0)
                                  REGION AT (302 662 0) TOO SMALL
                                  REJECTING (1003 317 0)
                                  LOOKING AT (1003 229 0)
                                  REGION AT (845 200 0) TOO SMALL
                                  REJECTING (409 130 0)
                                  LOOKING AT (400 130 0)
                                  REGION AT (400 109 0) TOO SMALL
                                  REJECTING (994 1064 0)
                                  LOOKING AT (600 1064 0)
                                  REGION AT (600 970 0) TOO SMALL
                                  LOOKING AT (268 581 0)
                                  FOUND REGION (155 554 0) TO (306 670 0)
. . . . . . . . . . . . . . . . . . . GOAL G-32 PUT BLOCK-1 (161 559 0)
. . . . . . . . . . . . . . . . . . . . GOAL G-33 GRASP BLOCK-1
. . . . . . . . . . . . . . . . . . . . . GOAL G-34 CLEAROFF BLOCK-1
                                         G-34 SUCCEEDS
                                       (7) GRASPING BLOCK-1
                                       G-33 SUCCEEDS
                                     (8) LIFTING BLOCK-1 FROM (356 770 800) TO (161
559 0)

                                     TAKING BLOCK-1 FROM STACK-12
                                     (9) LETTING GO OF BLOCK-1
                                     ADDING BLOCK-1 ON TABLE-1 (POS)
                                     G-32 SUCCEEDS
                                   G-31 SUCCEEDS
                                 G-14 SUCCEEDS
                               (10) MOVING HAND FROM (211 609 100) TO (406 820 800)
                               (11) GRASPING BLOCK-9
                               G-13 SUCCEEDS
                             (12) LIFTING BLOCK-9 FROM (306 720 600) TO (52 304 100)
                             TAKING BLOCK-9 FROM STACK-12
                             (13) LETTING GO OF BLOCK-9
                             ADDING BLOCK-9 ON BLOCK-0 (POS)
                             MAKING STACK STACK-15 BLOCK-9 BLOCK-0
                             G-12 SUCCEEDS
                           G-11 SUCCEEDS
                         G-10 SUCCEEDS
                       (14) MOVING HAND FROM (152 404 300) TO (406 820 600)
                       (15) GRASPING BLOCK-4
                       G-9 SUCCEEDS
                     (16) LIFTING BLOCK-4 FROM (306 720 400) TO (610 300 0)
                     TAKING BLOCK-4 FROM STACK-12
                     (17) LETTING GO OF BLOCK-4
                     ADDING BLOCK-4 ON TABLE-1 (POS)
                     G-8 SUCCEEDS
```

```
                G-7 SUCCEEDS
              G-6 SUCCEEDS
            (18) MOVING HAND FROM (710 400 200) TO (406 820 400)
            (19) GRASPING BLOCK-A
            G-5 SUCCEEDS
          MOVE TO (166 733 100) OVERLAPS BLOCK-A WITH BLOCK-1
          MOVE TO (166 733 100) OVERLAPS BLOCK-A WITH PYRAMID-1
          (20) LETTING GO OF BLOCK-A
          G-4 SUCCEEDS
        G-3 SUCCEEDS
. . GOAL G-35 GETRIDOF BLOCK-A
      LOOKING AT (827 585 0)
      REGION AT (810 580 0) TOO SMALL
      LOOKING AT (748 0 0)
      REGION AT (600 0 0) TOO SMALL
      LOOKING AT (340 529 0)
      REGION AT (302 429 0) TOO SMALL
      LOOKING AT (24 670 0)
      FOUND REGION (0 662 0) TO (306 974 0)
. . . GOAL G-36 PUT BLOCK-A (72 673 0)
. . . . GOAL G-37 GRASP BLOCK-A
. . . . . GOAL G-38 CLEAROFF BLOCK-A
            G-38 SUCCEEDS
          (21) GRASPING BLOCK-A
          G-37 SUCCEEDS
        (22) LIFTING BLOCK-A FROM (306 695 300) TO (72 673 0)
        TAKING BLOCK-A FROM STACK-12
        STACK-12 DISMANTLED
        (23) LETTING GO OF BLOCK-A
        ADDING BLOCK-A ON TABLE-1 (POS)
        G-36 SUCCEEDS
      G-35 SUCCEEDS
    G-2 SUCCEEDS
  (24) MOVING HAND FROM (172 790 100) TO (406 820 300)
  (25) GRASPING BLOCK-3
  G-1 SUCCEEDS
(26) LIFTING BLOCK-3 FROM (306 670 0) TO (306 670 900)
GT SUCCEEDS

REPLY (1 (OKAY))



CLEARTOP (BLOCK-1) (BLOCK-2) (BLOCK-3) (BLOCK-4) (BLOCK-5) (BLOCK-6) (BLOCK-7)
  (BLOCK-8) (BLOCK-9) (BLOCK-A) (PYRAMID-1) (PYRAMID-2) (PYRAMID-3) (PYRAMID-8)
GRASPING (HAND-1 BLOCK-3)
HASAV (BLOCK-0 SIZE LARGE POS) (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS)
  (BLOCK-2 COLOR GREEN POS) (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS)
  (BLOCK-3 SIZE LARGE POS) (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS)
  (BLOCK-5 COLOR BLUE POS) (BLOCK-5 SIZE LARGE POS) (BLOCK-6 COLOR BLACK POS)
  (BLOCK-6 SIZE LARGE POS) (BLOCK-7 COLOR BLACK POS) (BLOCK-7 SIZE LARGE POS)
  (BLOCK-8 COLOR BLACK POS) (BLOCK-8 SIZE LARGE POS) (BLOCK-9 COLOR BLACK POS)
  (BLOCK-9 SIZE LARGE POS) (BLOCK-A SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
  (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
  (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
  (PYRAMID-3 SIZE SMALL POS) (PYRAMID-8 SIZE LARGE POS)
HASREL (BLOCK-0 ON TABLE-1 POS) (BLOCK-1 ON TABLE-1 POS) (BLOCK-2 IN BOX-1 POS)
  (BLOCK-4 ON TABLE-1 POS) (BLOCK-5 ON TABLE-1 POS) (BLOCK-6 ON TABLE-1 POS)
  (BLOCK-7 ON TABLE-1 POS) (BLOCK-8 IN BOX-1 POS) (BLOCK-9 ON BLOCK-0 POS)
  (BLOCK-A ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (PYRAMID-1 ON TABLE-1 POS)
  (PYRAMID-2 ON TABLE-1 POS) (PYRAMID-3 ON TABLE-1 POS)
  (PYRAMID-8 ON TABLE-1 POS)
HASSIZE (BLOCK-0 300 300 100) (BLOCK-1 100 100 100) (BLOCK-2 200 200 200)
  (BLOCK-3 200 300 300) (BLOCK-4 200 200 200) (BLOCK-5 300 100 400)
  (BLOCK-6 200 200 200) (BLOCK-7 200 200 200) (BLOCK-8 200 200 200)
  (BLOCK-9 200 200 200) (BLOCK-A 200 250 100) (BOX-1 600 600 1)
  (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 240)
  (PYRAMID-8 400 220 100) (TABLE-1 1200 1200 0)
INSTACK (BLOCK-0 STACK-15) (BLOCK-9 STACK-15)
ISA (BLOCK-0 BLOCK) (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK)
  (BLOCK-4 BLOCK) (BLOCK-5 BLOCK) (BLOCK-6 BLOCK) (BLOCK-7 BLOCK)
  (BLOCK-8 BLOCK) (BLOCK-9 BLOCK) (BLOCK-A BLOCK) (BOX-1 BOX) (HAND-1 HAND)
  (PYRAMID-1 PYRAMID) (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID)
  (PYRAMID-8 PYRAMID) (TABLE-1 TABLE)
LOCAT (BLOCK-0 2 254 0) (BLOCK-1 161 559 0) (BLOCK-2 800 600 1)
  (BLOCK-3 306 670 900) (BLOCK-4 610 300 0) (BLOCK-5 545 216 0)
  (BLOCK-6 100 0 0) (BLOCK-7 400 0 0) (BLOCK-8 1000 600 1) (BLOCK-9 52 304 100)
  (BLOCK-A 72 673 0) (BOX-1 600 600 0) (HAND-1 406 820 1200)
  (PYRAMID-1 55 562 0) (PYRAMID-2 800 229 0) (PYRAMID-3 300 9 0)
  (PYRAMID-8 183 974 0) (TABLE-1 0 0 0)

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```





```
RUN TIME 39 MIN, 3.19 SEC

EXAM     TRY     FIPE    WMACT   E/F     E/T     T/F
6494     1983    1290    4019    5.41    3.27    1.65
0.361    1.10    1.95    0.406   SEC AVG

2716 INSERTS 2103 DELETES 626 WARNINGS 24 NEW OBJECTS
MAX :SMPX LENGTH 172
CORE (FREE.FULL): (5728 . 1209) USED (9991 . 1776)
FIRED 91 OUT OF 400 PRODS

                 - - - - - - - - - - - - - - - - - - - -


NINTH SEGMENT: TRYING TO GET BACKUP WITH A DIFFERENCE

ADDING SIZE LAPGE (POS) TO PYPAMID-B
ADDING PYPAMID PYPAMID-B
22 INPUT TEXT IS " PUT IT DOWN "
OBJ-1 REFERS PYPAMID-B
STARTING GT PUT PYPAMID-B DOWN
GOAL G-1 GETRIDOF PYPAMID-B
LOOKING AT (902 155 0)
REGION AT (845 109 0) TOO SMALL
LOOKING AT (650 595 0)
REGION AT (600 554 0) TOO SMALL
REJECTING (620 263 0)
LOOKING AT (620 216 0)
REGION AT (600 200 0) TOO SMALL
REJECTING (870 1049 0)
LOOKING AT (600 1049 0)
REGION AT (600 970 0) TOO SMALL
LOOKING AT (869 224 0)
REGION AT (845 200 0) TOO SMALL
LOOKING AT (138 953 0)
REGION AT (0 554 0) TOO SMALL
REJECTING (369 979 0)
LOOKING AT (369 970 0)
REGION AT (302 970 0) TOO SMALL
LOOKING AT (91 24 0)
REGION AT (0 0 0) TOO SMALL
LOOKING AT (103 1053 0)
FOUND PEGION (0 970 0) TO (600 1200 0)
. GOAL G-2 PUT PYPAMID-B (15 900 0)
. . GOAL G-3 GPASP PYPAMID-B
      G-3 SUCCEEDS
    (0) LIFTING PYPAMID-B FROM (495 156 300) TO (15 900 0)
    (1) LETTING GO OF PYPAMID-B
    ADDING PYPAMID-B ON TABLE-1 (POS)
    G-2 SUCCEEDS
  G-1 SUCCEEDS
GT SUCCEEDS

REPLY (1 (OKAY))
```

```
24 INPUT TEXT IS " PICK UP THE LARGE RED BLOCK "
OBJ-1 AMBIG L4-1 BLOCK-0 BLOCK-2 ...
OBJ-1 REFERS BLOCK-3
STARTING GT PICKUP BLOCK-3
GOAL G-1 GRASP BLOCK-3
. GOAL G-2 CLEAROFF BLOCK-3
. . GOAL G-3 GETRIDOF BLOCK-A
. . . GOAL G-4 GRASP BLOCK-A
. . . . GOAL G-5 CLEAROFF BLOCK-A
. . . . . GOAL G-6 GETRIDOF BLOCK-4
. . . . . . GOAL G-7 GRASP BLOCK-4
. . . . . . . GOAL G-8 CLEAROFF BLOCK-4
. . . . . . . . GOAL G-9 GETRIDOF BLOCK-9
. . . . . . . . . GOAL G-10 GRASP BLOCK-9
. . . . . . . . . . GOAL G-11 CLEAROFF BLOCK-9
. . . . . . . . . . . GOAL G-12 GETRIDOF BLOCK-1
. . . . . . . . . . . . GOAL G-13 GRASP BLOCK-1
. . . . . . . . . . . . . GOAL G-14 CLEAROFF BLOCK-1
. . . . . . . . . . . . . . GOAL G-15 GETRIDOF PYRAMID-1
. . . . . . . . . . . . . . . GOAL G-16 GRASP PYRAMID-1
. . . . . . . . . . . . . . . . GOAL G-17 CLEAROFF PYRAMID-1
                                    G-17 SUCCEEDS
                                  (0) MOVING HAND FROM (215 1090 100) TO (406 820
 1000)
                                  (1) GRASPING PYRAMID-1
                                  G-16 SUCCEEDS
                                LOOKING AT (556 905 0)
                                REGION AT (506 970 0) TOO SMALL
                                REJECTING (948 002 0)
                                LOOKING AT (948 600 0)
                                REGION AT (845 554 0) TOO SMALL
                                LOOKING AT (405 511 0)
                                FOUND REGION (400 429 0) TO (600 600 0)
. . . . . . . . . . . . . . . . GOAL G-18 PUT PYRAMID-1 (467 448 0)
. . . . . . . . . . . . . . . . . GOAL G-19 GRASP PYRAMID-1
                                    G-19 SUCCEEDS
                                  (2) LIFTING PYRAMID-1 FROM (356 770 900) TO (
 467 448 0)
                                  TAKING PYRAMID-1 FROM STACK-12
                                  ADDING PYRAMID-1 ON TABLE-1 (POS)
                                  (3) LETTING GO OF PYRAMID-1
                                  G-18 SUCCEEDS
                                G-15 SUCCEEDS
                              G-14 SUCCEEDS
                            (4) MOVING HAND FROM (517 490 100) TO (406 820 900)
                            (5) GRASPING BLOCK-1
                            G-13 SUCCEEDS
                          REJECTING (203 163 0)
                          LOOKING AT (203 200 0)
                          REGION AT (0 200 0) TOO SMALL
                          REJECTING (214 507 0)
                          LOOKING AT (214 554 0)
                          FOUND REGION (0 554 0) TO (306 670 0)
. . . . . . . . . . . . . GOAL G-20 PUT BLOCK-1 (154 559 0)
. . . . . . . . . . . . . . GOAL G-21 GRASP BLOCK-1
                              G-21 SUCCEEDS
                            (6) LIFTING BLOCK-1 FROM (358 770 800) TO (154 559 0
                            TAKING BLOCK-1 FROM STACK-12
                          (7) LETTING GO OF BLOCK-1
                          ADDING BLOCK-1 ON TABLE-1 (POS)
                          G-20 SUCCEEDS
                        G-12 SUCCEEDS
                      G-11 SUCCEEDS
                    (8) MOVING HAND FROM (204 609 100) TO (406 820 800)
                    (9) GRASPING BLOCK-9
                    G-10 SUCCEEDS
                  LOOKING AT (460 995 0)
                  REGION AT (415 970 0) TOO SMALL
                  REJECTING (420 682 0)
                  LOOKING AT (420 670 0)
                  REGION AT (415 659 0) TOO SMALL
                  REJECTING (1057 945 0)
                  LOOKING AT (1057 600 0)
                  REGION AT (845 554 0) TOO SMALL
                  REJECTING (930 348 0)
                  LOOKING AT (800 348 0)
                  REGION AT (845 316 0) TOO SMALL
                  REJECTING (165 1013 0)
                  LOOKING AT (165 900 0)
                  REGION AT (0 970 0) TOO SMALL
                  REJECTING (612 1051 0)
                  FINDSPACE LIMIT EXCEEDED
. . . . . . . . . GOAL G-22 GRASP BLOCK-9
                    G-22 SUCCEEDS
                  FOUND REGION CLEARTOP BLOCK-0
. . . . . . . . . GOAL G-23 PUT BLOCK-9 (52 304 100)
. . . . . . . . . . GOAL G-24 GRASP BLOCK-9
                      G-24 SUCCEEDS
                    (10) LIFTING BLOCK-9 FROM (306 720 600) TO (52 304 100)
                    TAKING BLOCK-9 FROM STACK-12
                    (11) LETTING GO OF BLOCK-9
                    ADDING BLOCK-9 ON BLOCK-0 (POS)
                    MAKING STACK STACK-13 BLOCK-9 BLOCK-0
                    G-23 SUCCEEDS
                  G-9 SUCCEEDS
                G-8 SUCCEEDS
              (12) MOVING HAND FROM (152 404 300) TO (406 820 600)
              (13) GRASPING BLOCK-4
              G-7 SUCCEEDS
            LOOKING AT (359 432 0)
            REGION AT (302 429 0) TOO SMALL
            LOOKING AT (736 205 0)
            REGION AT (600 200 0) TOO SMALL
            REJECTING (1049 423 0)
            LOOKING AT (1049 429 0)
            REGION AT (845 429 0) TOO SMALL
            LOOKING AT (40 922 0)
            FOUND REGION (0 659 0) TO (306 900 0)
. . . . . . GOAL G-25 PUT BLOCK-4 (14 702 0)
. . . . . . . GOAL G-26 GRASP BLOCK-4
                G-26 SUCCEEDS
              (14) LIFTING BLOCK-4 FROM (306 720 400) TO (14 702 0)
              TAKING BLOCK-4 FROM STACK-12
              (15) LETTING GO OF BLOCK-4
              ADDING BLOCK-4 ON TABLE-1 (POS)
              G-25 SUCCEEDS
            G-6 SUCCEEDS
          G-5 SUCCEEDS
        (16) MOVING HAND FROM (114 802 200) TO (406 820 400)
        (17) GRASPING BLOCK-A
        G-4 SUCCEEDS
      LOOKING AT (135 606 0)
      REGION AT (0 554 0) TOO SMALL
      REJECTING (739 963 0)
      LOOKING AT (600 963 0)
      REGION AT (600 902 0) TOO SMALL
      REJECTING (95 418 0)
      LOOKING AT (2 418 0)
      REGION AT (0 316 0) TOO SMALL
      REJECTING (171 348 0)
      LOOKING AT (171 254 0)
      REGION AT (0 200 0) TOO SMALL
      REJECTING (257 594 0)
      LOOKING AT (254 594 0)
      REGION AT (254 554 0) TOO SMALL
      LOOKING AT (567 833 0)
      REGION AT (567 659 0) TOO SMALL
      FINDSPACE LIMIT EXCEEDED
. . . GOAL G-27 GRASP BLOCK-A
        G-27 SUCCEEDS
      LOOKING AT (23 354 100)
```

```
     REGION AT (2 254 100) TOO SMALL
     LOOKING AT (76 260 100)
     REGION AT (2 254 100) TOO SMALL
     LOOKING AT (160 294 100)
     REGION AT (2 254 100) TOO SMALL
     REJECTING (71 363 100)
     LOOKING AT (52 363 100)
     REGION AT (2 254 100) TOO SMALL
     LOOKING AT (5 307 100)
     REGION AT (2 254 100) TOO SMALL
     REJECTING (140 320 100)
     LOOKING AT (140 304 100)
     REGION AT (2 254 100) TOO SMALL
     REJECTING (77 360 100)
     LOOKING AT (52 360 100)
     REGION AT (2 254 100) TOO SMALL
     FINDSPACE LIMIT EXCEEDED
     G-3 EXHAUSTED
     (17) LETTING GO OF BLOCK-A
     (16) MOVING HAND FROM (406 820 400) TO (114 802 200)
     (15) GRASPING BLOCK-4
     (14) LIFTING BLOCK-4 FROM (14 702 0) TO (306 720 400)
     ADDING BLOCK-4 ON BLOCK-A (POS)
     ADDING BLOCK-4 TO STACK-12
. . . . . GOAL G-25 RETRY GETRIDOF BLOCK-4
. . . . . . GOAL G-28 GRASP BLOCK-4
              G-28 SUCCEEDS
            LOOKING AT (461 547 0)
            REGION AT (415 429 0) TOO SMALL
            LOOKING AT (543 373 0)
            REGION AT (506 316 0) TOO SMALL
            LOOKING AT (816 573 0)
            REGION AT (600 554 0) TOO SMALL
            LOOKING AT (630 510 0)
            REGION AT (600 429 0) TOO SMALL
            LOOKING AT (615 140 0)
            REGION AT (600 109 0) TOO SMALL
            REJECTING (90 327 0)
            LOOKING AT (90 254 0)
            REGION AT (0 200 0) TOO SMALL
            REJECTING (270 1031 0)
            LOOKING AT (270 900 0)
            REGION AT (254 970 0) TOO SMALL
            LOOKING AT (971 197 0)
            REGION AT (845 109 0) TOO SMALL
            FINDSPACE LIMIT EXCEEDED
. . . . . . GOAL G-29 GRASP BLOCK-4
              G-29 SUCCEEDS
            REJECTING (92 376 100)
            LOOKING AT (52 376 100)
            REGION AT (2 254 100) TOO SMALL
            LOOKING AT (0 201 100)
            REGION AT (2 254 100) TOO SMALL
            LOOKING AT (169 259 100)
            REGION AT (2 254 100) TOO SMALL
            REJECTING (89 330 100)
            LOOKING AT (89 304 100)
            REGION AT (2 254 100) TOO SMALL
            REJECTING (72 419 100)
            LOOKING AT (52 419 100)
            REGION AT (2 254 100) TOO SMALL
            LOOKING AT (50 412 100)
            REGION AT (2 254 100) TOO SMALL
            REJECTING (142 365 100)
            FINDSPACE LIMIT EXCEEDED
. . . . . . GOAL G-30 GRASP BLOCK-4
              G-30 SUCCEEDS
            FOUND REGION CLEARTOP BLOCK-2
. . . . . . GOAL G-31 PUT BLOCK-4 (800 600 201)
. . . . . . . GOAL G-32 GRASP BLOCK-4
                G-32 SUCCEEDS
              (14) LIFTING BLOCK-4 FROM (306 720 400) TO (800 600 201)
              TAKING BLOCK-4 FROM STACK-12
              (15) LETTING GO OF BLOCK-4
              ADDING BLOCK-4 ON BLOCK-2 (POS)
              MAKING STACK STACK-14 BLOCK-4 BLOCK-2
              G-31 SUCCEEDS
            G-25 SUCCEEDS
          G-6 SUCCEEDS
        G-5 SUCCEEDS
      (16) MOVING HAND FROM (900 700 401) TO (406 820 400)
      (17) GRASPING BLOCK-A
      G-4 SUCCEEDS
```

```
        LOOKING AT (85 80 0)
        REGION AT (0 0 0) TOO SMALL
        REJECTING (960 706 0)
        LOOKING AT (960 600 0)
        REGION AT (845 554 0) TOO SMALL
        REJECTING (510 159 0)
        LOOKING AT (510 200 0)
        REGION AT (506 200 0) TOO SMALL
        REJECTING (729 930 0)
        LOOKING AT (600 930 0)
        REGION AT (600 659 0) TOO SMALL
        REJECTING (90 443 0)
        LOOKING AT (2 443 0)
        REGION AT (0 429 0) TOO SMALL
        REJECTING (412 771 0)
        FINDSPACE LIMIT EXCEEDED
  . . . GOAL G-33 GRASP BLOCK-A
          G-33 SUCCEEDS
        LOOKING AT (42 376 100)
        REGION AT (2 254 100) TOO SMALL
        LOOKING AT (89 294 100)
        REGION AT (2 254 100) TOO SMALL
        REJECTING (80 333 100)
        LOOKING AT (80 304 100)
        REGION AT (2 254 100) TOO SMALL
        LOOKING AT (82 281 100)
        REGION AT (2 254 100) TOO SMALL
        REJECTING (87 314 100)
        LOOKING AT (87 304 100)
        REGION AT (2 254 100) TOO SMALL
        LOOKING AT (103 263 100)
        REGION AT (2 254 100) TOO SMALL
        REJECTING (106 318 100)
        LOOKING AT (106 304 100)
        REGION AT (2 254 100) TOO SMALL
        FINDSPACE LIMIT EXCEEDED
        G-3 EXHAUSTED
        (17) LETTING GO OF BLOCK-A
        (16) MOVING HAND FROM (406 820 400) TO (900 700 401)
        (15) GRASPING BLOCK-4
        (14) LIFTING BLOCK-4 FROM (800 600 201) TO (306 720 400)
        TAKING BLOCK-4 FROM STACK-14
        STACK-14 DISMANTLED
        ADDING BLOCK-4 ON BLOCK-A (POS)
        ADDING BLOCK-4 TO STACK-12
  . . . . . . . GOAL G-25 RETRY GETRIDOF BLOCK-4
  . . . . . . . . GOAL G-34 GRASP BLOCK-4
                    G-34 SUCCEEDS
                  REJECTING (342 767 300)
                  LOOKING AT (342 695 300)
                  REGION AT (306 670 300) TOO SMALL
                  REJECTING (319 706 300)
                  LOOKING AT (319 695 300)
                  REGION AT (306 670 300) TOO SMALL
                  REJECTING (335 736 300)
                  LOOKING AT (335 695 300)
                  REGION AT (306 670 300) TOO SMALL
                  LOOKING AT (306 834 300)
                  REGION AT (306 670 300) TOO SMALL
                  REJECTING (345 737 300)
                  LOOKING AT (345 695 300)
                  REGION AT (306 670 300) TOO SMALL
                  REJECTING (311 781 300)
                  FINDSPACE LIMIT EXCEEDED
  . . . . . . . . GOAL G-35 GRASP BLOCK-4
                    G-35 SUCCEEDS
                  FOUND REGION CLEARTOP BLOCK-6
  . . . . . . . . GOAL G-36 PUT BLOCK-4 (100 0 200)
  . . . . . . . . . GOAL G-37 GRASP BLOCK-4
                      G-37 SUCCEEDS
                    (14) LIFTING BLOCK-4 FROM (306 720 400) TO (100 0 200)
                    TAKING BLOCK-4 FROM STACK-12
                    (15) LETTING GO OF BLOCK-4
                    ADDING BLOCK-4 ON BLOCK-6 (POS)
                    MAKING STACK STACK-15 BLOCK-4 BLOCK-6
                    G-36 SUCCEEDS
                  G-25 SUCCEEDS
                G-6 SUCCEEDS
              G-5 SUCCEEDS
            (16) MOVING HAND FROM (200 100 400) TO (406 820 400)
            (17) GRASPING BLOCK-A
          G-4 SUCCEEDS
        REJECTING (960 624 0)
```

```
          LOOKING AT (960 600 0)
          REGION AT (845 554 0) TOO SMALL
          LOOKING AT (130 856 0)
          FOUND REGION (0 659 0) TO (306 900 0)
    . . . GOAL G-30 PUT BLOCK-A (25 702 0)
    . . . . GOAL G-39 GRASP BLOCK-A
              G-39 SUCCEEDS
            (18) LIFTING BLOCK-A FROM (306 806 300) TO (25 702 0)
            TAKING BLOCK-A FROM STACK-12
            STACK-12 DISMANTLED
            (19) LETTING GO OF BLOCK-A
            ADDING BLOCK-A ON TABLE-1 (POS)
            G-30 SUCCEEDS
          G-3 SUCCEEDS
        G-2 SUCCEEDS
      (20) MOVING HAND FROM (125 827 100) TO (406 820 300)
      (21) GRASPING BLOCK-3
     G-1 SUCCEEDS
    (22) LIFTING BLOCK-3 FROM (306 670 0) TO (306 670 900)
    GT SUCCEEDS

    REPLY (1 (OKAY))


    CLEARTOP (BLOCK-1) (BLOCK-2) (BLOCK-3) (BLOCK-4) (BLOCK-5) (BLOCK-7) (BLOCK-8)
      (BLOCK-9) (BLOCK-A) (PYRAMID-1) (PYRAMID-2) (PYRAMID-3) (PYRAMID-8)
    GRASPING (HAND-1 BLOCK-3)
    HASAV (BLOCK-0 SIZE LARGE POS) (BLOCK-1 COLOR RED POS) (BLOCK-1 SIZE SMALL POS)
      (BLOCK-2 COLOR GREEN POS) (BLOCK-2 SIZE LARGE POS) (BLOCK-3 COLOR RED POS)
      (BLOCK-3 SIZE LARGE POS) (BLOCK-4 COLOR GREEN POS) (BLOCK-4 SIZE LARGE POS)
      (BLOCK-5 COLOR BLUE POS) (BLOCK-5 SIZE LARGE POS) (BLOCK-6 COLOR BLACK POS)
      (BLOCK-6 SIZE LARGE POS) (BLOCK-7 COLOR BLACK POS) (BLOCK-7 SIZE LARGE POS)
      (BLOCK-8 COLOR BLACK POS) (BLOCK-8 SIZE LARGE POS) (BLOCK-9 COLOR BLACK POS)
      (BLOCK-9 SIZE LARGE POS) (BLOCK-A SIZE LARGE POS) (PYRAMID-1 COLOR GREEN POS)
      (PYRAMID-1 SIZE SMALL POS) (PYRAMID-2 COLOR BLUE POS)
      (PYRAMID-2 SIZE LARGE POS) (PYRAMID-3 COLOR RED POS)
      (PYRAMID-3 SIZE SMALL POS) (PYRAMID-8 SIZE LARGE POS)
    HASREL (BLOCK-0 ON TABLE-1 POS) (BLOCK-1 ON TABLE-1 POS) (BLOCK-2 IN BOX-1 POS)
      (BLOCK-4 ON BLOCK-6 POS) (BLOCK-5 ON TABLE-1 POS) (BLOCK-6 ON TABLE-1 POS)
      (BLOCK-7 ON TABLE-1 POS) (BLOCK-8 IN BOX-1 POS) (BLOCK-9 ON BLOCK-0 POS)
      (BLOCK-A ON TABLE-1 POS) (BOX-1 ON TABLE-1 POS) (PYRAMID-1 ON TABLE-1 POS)
      (PYRAMID-2 ON TABLE-1 POS) (PYRAMID-3 ON TABLE-1 POS)
      (PYRAMID-8 ON TABLE-1 POS)
    HASSIZE (BLOCK-0 300 300 100) (BLOCK-1 100 100 100) (BLOCK-2 200 200 200)
      (BLOCK-3 200 300 300) (BLOCK-4 200 200 200) (BLOCK-5 300 100 400)
      (BLOCK-6 200 200 200) (BLOCK-7 200 200 200) (BLOCK-8 200 200 200)
      (BLOCK-9 200 200 200) (BLOCK-A 200 250 100) (BOX-1 600 600 1)
      (PYRAMID-1 100 100 100) (PYRAMID-2 300 200 200) (PYRAMID-3 100 100 240)
      (PYRAMID-8 400 220 100) (TABLE-1 1200 1200 0)
    INSTACK (BLOCK-0 STACK-13) (BLOCK-4 STACK-15) (BLOCK-6 STACK-15)
      (BLOCK-9 STACK-13)
    ISA (BLOCK-0 BLOCK) (BLOCK-1 BLOCK) (BLOCK-2 BLOCK) (BLOCK-3 BLOCK)
      (BLOCK-4 BLOCK) (BLOCK-5 BLOCK) (BLOCK-6 BLOCK) (BLOCK-7 BLOCK)
      (BLOCK-8 BLOCK) (BLOCK-9 BLOCK) (BLOCK-A BLOCK) (BOX-1 BOX) (HAND-1 HAND)
      (PYRAMID-1 PYRAMID) (PYRAMID-2 PYRAMID) (PYRAMID-3 PYRAMID)
      (PYRAMID-8 PYRAMID) (TABLE-1 TABLE)
    LOCAT (BLOCK-0 2 254 0) (BLOCK-1 154 559 0) (BLOCK-2 800 600 1)
      (BLOCK-3 306 670 900) (BLOCK-4 100 0 200) (BLOCK-5 545 216 0)
      (BLOCK-6 100 0 0) (BLOCK-7 400 0 0) (BLOCK-8 1000 600 1) (BLOCK-9 52 304 100)
      (BLOCK-A 25 702 0) (BOX-1 600 600 0) (HAND-1 406 820 1200)
      (PYRAMID-1 467 440 0) (PYRAMID-2 800 229 0) (PYRAMID-3 300 9 0)
      (PYRAMID-8 15 900 0) (TABLE-1 0 0 0)

    ..............................................................................
    .
    .
    .
    L+P8
    .
    .
    .                         ++H1
    .
    L+BA              LRB3
    .                                     ++X1        LGB2          LB08
    .        SRB1
    .
    .                           SGP1
    .
    . LB09
    L+B0
    .                                 LB06              LBP2
    .
    .     LGB4
```

```
.     L886          SRP3  L887
......................................................................

RUN TIME 24 MIN. 47.9 SEC

EXAM     TRY      FIRE     WMACT   E/F     E/T     T/F
4152     1518     942      3893    4.41    2.74    1.61
0.358    0.980    1.58     0.403   SEC AVG

2056 INSERTS 1598 DELETES 517 WARNINGS 25 NEW OBJECTS
MAX :SMPX LENGTH 174
CORE (FREE.FULL): (7468 . 1542) USED (8253 . 1444)
FIRED 93 OUT OF 410 PRODS
```

Appendix I. DETAILED TRACE FOR WBLOX TEST1

```
ASSERT (TEST1 T

TOP LEVEL ASSERT (TEST1 (QUOTE T))
INSERTING (TEST1 T)    V1/

11.  V1-1
USING (TEST1 T)
INSERTING (SCANFIN LE-1) (SENTENCE S-1) (ENDMARK LE-1) (ENDMARK RE-1)
 (TEXT 1 (PUT THE SMALL RED BLOCK ON THE BLUE BLOCK)) (LEFTOF LE-1 P1-1)
 (EQPUT P1-1) (LEFTOF P1-1 T2-1) (EQTHE T2-1) (LEFTOF T2-1 S3-1) (EQSMALL S3-1)
 (LEFTOF S3-1 R4-1) (EQRED R4-1) (LEFTOF R4-1 B5-1) (EQBLOCK B5-1)
 (LEFTOF B5-1 O6-1) (EQON O6-1) (LEFTOF O6-1 T7-1) (EQTHE T7-1)
 (LEFTOF T7-1 B8-1) (EQBLUE B8-1) (LEFTOF B8-1 B9-1) (EQBLOCK B9-1)
 (LEFTOF B9-1 RE-1)    S0/

12.  S0-1   "SCAN LE"
USING (SCANFIN LE-1) (ENDMARK LE-1) (LEFTOF LE-1 P1-1)
 (TEXT 1 (PUT THE SMALL RED BLOCK ON THE BLUE BLOCK))

TRACING
1 INPUT TEXT IS " PUT THE SMALL RED BLOCK ON THE BLUE BLOCK "


WARNING (T) ALREADY UNDER TRACING  =.
INSERTING (SCAN P1-1) (SCANFIN P1-1) (NOT (SCANFIN LE-1)) (NREPLY 0) (TRACING T)
     G1G2/G44/

13.  G44-1   "PUT INIT"
USING (SCAN P1-1) (EQPUT P1-1) (SENTENCE S-1)
INSERTING (IMPTYPE S-1 PUT) (WORDEQ P1-1 PUT) (EXPECTMOD S-1 DOWN)
 (EXPECTMOD S-1 IN) (EXPECTMOD S-1 ON) (ISIMPER P1-1) (GTYPED S-1) (GS1 S-1)
 (NOT (SCAN P1-1)) (NOT (EQPUT P1-1))  M81FM86FM87M88M85M84FM84M83PM83M82PM82
 M81G1E8E4M8AM81M82M84M84AF34F341T71T72M8EG8G1M85M88G51G49B9818988181M71M6354/
 S1/

14.  S1-1   "SCAN ON"
USING (SCANFIN P1-1) (LEFTOF P1-1 T2-1)
INSERTING (SCAN T2-1) (SCANFIN T2-1) (NOT (SCANFIN P1-1))      G7G9G21G41G42
 G43T7/T13/T27/T34/T71T72T82T81T97T44/G2/G44G1/

TRACING
OBJ-2 REFERS BLOCK-5


WARNING (T) ALREADY UNDER TRACING  =.
INSERTING (REFERS OBJ-2 BLOCK-5) (TRACING T) (NOT (OCHK OBJ-2 B9-1))
 (NOT (FINDPOSS OBJ-2 BLOCK-5))       B341B34B33IB33F29/F23B98IB93B81B28B28
 B298I98IB815IB1583/B1/

187.  B1-1   "DEF REF"
USING (REFERS OBJ-2 BLOCK-5) (CUROBJ OBJ-2 OBJ-1) (HASRELN OBJ-1 ON POS)
 (EARREF OBJ-1 B5-1)
INSERTING (RELRESTRCHK OBJ-1 B5-1 ON BLOCK-5 POS) (CUROBJP OBJ-2 OBJ-1)
 (OLDREF OBJ-2)       B101/

188.  B101-1  "REL RESTR IND"
USING (RELRESTRCHK OBJ-1 B5-1 ON BLOCK-5 POS) (ISINDREL ON)
INSERTING (CHAINREL ON BLOCK-5 ON BLOCK-5)
 (RELRESTRCHK2 OBJ-1 B5-1 ON BLOCK-5 POS)
 (NOT (RELRESTRCHK OBJ-1 B5-1 ON BLOCK-5 POS)) B10L/B10K/B10J/B13B13IB17B1BC
 B1BIB18/B18/

189.  B18-1   "REL RCHK INC"
USING (RELRESTRCHK2 OBJ-1 B5-1 ON BLOCK-5 POS) (REFERS OBJ-1 BLOCK-1)
INSERTING (RELINCONT OBJ-1 B5-1 ON BLOCK-5 POS)
 (NOT (RELRESTRCHK2 OBJ-1 B5-1 ON BLOCK-5 POS))       E31/

190.  E31-1   "TRACE R INC"
USING (RELINCONT OBJ-1 B5-1 ON BLOCK-5 POS)

TRACING
RELINCON OBJ-1 B5-1 ON BLOCK-5 POS


WARNING (T) ALREADY UNDER TRACING  =.
INSERTING (RELINCON OBJ-1 B5-1 ON BLOCK-5 POS) (TRACING T)     M1M11/M81/M81/

191.  M81-1   "REL INCON IMP"
USING (RELINCON OBJ-1 B5-1 ON BLOCK-5 POS) (GS1 S-1) (CUROBJP OBJ-1 MAIN)
 (EXPECTMOD S-1 ON)
```

```
INSERTING (IMPREL S-1 ON BLOCK-5) (NOT (REL INCON OBJ-1 B9-1 ON BLOCK-5 POS))
  M82M84FM84M83PM83M82PM82M71M66B38B39B43F34F34IB56B46B44B36B34B34IM12M53B33
  B33IB46B59IB55B53B45B2SB3/M18M15M5M2V48M63V37V38V39V32V31V30V19V17V10B57E8
  B58FP1R2N31N9AG5E6E4S1/B4/

I 92.  B4-1   "SCAN FIN"
USING (SCANFIN B9-1) (LEFTOF B9-1 RE-1) (ENDMARK RE-1) (SENTENCE S-1)
INSERTING (NPBOUND RE-1) (SENTBOUND S-1) (NOT (SCANFIN B9-1))   B56B58B57B55/

I 93.  B55-1   "NPEND REDO"
USING (NPBOUND RE-1) (CUROBJP OBJ-1 MAIN)
INSERTING (CUROBJ OBJ-1 MAIN)   B56B43V14B33B33IF53N1/N9AM53/A1/R12/R11A5N31N5
  N3F61F41B51/

I 94.  B51-1   "NPEND UNDO"
USING (NPBOUND RE-1) (CUROBJ OBJ-2 OBJ-1) (REFERS OBJ-2 BLOCK-5)
INSERTING (NOT (CUROBJ OBJ-2 OBJ-1))   B56B44B34B34IB55/B48B38B24B14B1/M71/

I 95.  M71-1   "IMP OBJ"
USING (SENTBOUND S-1) (GS1 S-1) (IMPREL S-1 ON BLOCK-5) (CUROBJ OBJ-1 MAIN)
  (REFERS OBJ-1 BLOCK-1)
INSERTING (IMPOBJ S-1 BLOCK-1)  M66M81/M82/

I 96.  M82-1   "CMD PUTON"
USING (SENTBOUND S-1) (IMPTYPE S-1 PUT) (IMPOBJ S-1 BLOCK-1)
  (IMPREL S-1 ON BLOCK-5) (ISA BLOCK-5 BLOCK)
INSERTING (WBPINIT GT) (PUTON GT BLOCK-1 BLOCK-5)
  (CHECKPUTON BLOCK-1 ON BLOCK-5)      M89/

I 97.  M89-1   "WB P INIT"
USING (WBPINIT GT) (SENTBOUND S-1)
INSERTING (EVENTTIME 0) (CHOICECOUNT 0) (HASLEVEL GT 0) (NOT (WBPINIT GT))
  (NOT (SENTBOUND S-1)) W33W34W35W36W47W45W53W54W56W57W12W17W20/W22W25Q49Q47U
  Q47Q2Q1W22BW58W52BW51W46W43W42BW32W31W30W53AW54AW54XW14W140W16W26W26XW2
  W19W18W15W13W55W53OQ45Q43Q31Q82Q81W27MW24FW24W23FW10W4W3W1W0TW0SW0GW0FW0
  W38W23/

I 98.  W23-1   "PUT ON 1"
USING (PUTON GT BLOCK-1 BLOCK-5)

TRACING
STARTING GT PUTON BLOCK-1 ONTO BLOCK-5


WARNING (T) ALREADY UNDER TRACING  =.
INSERTING (PUTON1 GT BLOCK-1 BLOCK-5) (NOT (PUTON GT BLOCK-1 BLOCK-5))
  (NEXTF GT (FAILPUTON1 GT BLOCK-1 BLOCK-5)) (TRACING T)      W26XW24F/W24/

I 99.  W24-1   "PUT ON"
USING (PUTON1 GT BLOCK-1 BLOCK-5) (HASSIZE BLOCK-1 100 100 100) (HASLEVEL GT 0)
  (HASSIZE BLOCK-5 300 100 400)

TRACING
GOAL G-1 CLEAROFF BLOCK-1


WARNING (T) ALREADY UNDER TRACING  =.
INSERTING (CLEAROFF G-1 BLOCK-1)
  (NEXT G-1 (FINDSPACE BLOCK-5 BLOCK-1 100 100 100)) (HASLEVEL G-1 1)
  (PUTONPUT GT BLOCK-1 BLOCK-5) (TRACING T) (NOT (PUTON1 GT BLOCK-1 BLOCK-5))
  W3/

I 100.  W3-1   "CLEAR OFF"
USING (CLEAROFF G-1 BLOCK-1) (HASREL PYRAMID-1 ON BLOCK-1 POS)
  (HASSIZE PYRAMID-1 100 100 100) (HASLEVEL G-1 1)

TRACING
. GOAL G-2 GETRIDOF PYRAMID-1


WARNING (T) ALREADY UNDER TRACING  =.
INSERTING (GETRIDOF G-2 PYRAMID-1) (HASLEVEL G-2 2)
  (NEXT G-2 (CLEAROFF G-1 BLOCK-1)) (NOT (CLEAROFF G-1 BLOCK-1)) (TRACING T)
  W18W15W15W11/

I 101.  W11-1   "GET RID OF START"
USING (GETRIDOF G-2 PYRAMID-1) (ISA TABLE-1 TABLE)
  (HASSIZE PYRAMID-1 100 100 100)
INSERTING (FINDSPACE TABLE-1 PYRAMID-1 100 100 100)
  (GETRIDPUT G-2 PYRAMID-1 TABLE-1) (NOT (GETRIDOF G-2 PYRAMID-1))      Q54/

I 102.  Q54-1   "FIND RANDOM"
USING (FINDSPACE TABLE-1 PYRAMID-1 100 100 100) (ISA TABLE-1 TABLE)
INSERTING (LOCATESPACE TABLE-1 PYRAMID-1 100 100 100)
  (USERESULT TABLE-1 PYRAMID-1 100 100 RANDOM)
  (NOT (FINDSPACE TABLE-1 PYRAMID-1 100 100 100))      Q57F/Q61/

I 103.  Q61-1   "LOCATE START"
USING (LOCATESPACE TABLE-1 PYRAMID-1 100 100 100) (LOCAT TABLE-1 0 0 0)
  (HASSIZE TABLE-1 1200 1200 0)
INSERTING (FINDLOWPAIR 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)
  (NOT (LOCATESPACE TABLE-1 PYRAMID-1 100 100 100))    Q63/Q64BQ64AQ64/Q62/

I 104.  Q62-1   "LOW PAIR"
USING (FINDLOWPAIR 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)

TRACING
 LOOKING AT (820 373 0)


WARNING (T) ALREADY UNDER TRACING  =.
INSERTING (FINDLOWX PYRAMID-1 0 820 0 1200 0)
  (FINDLOWY PYRAMID-1 0 1200 0 373 0) (LOWX 10 PYRAMID-1 0)
  (LOWY 10 PYRAMID-1 0)
  (GROWTOFITO 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)
  (NOT (FINDLOWPAIR 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100))
  (TRACING T)   Q68Q69Q65/

I 105.  Q65-1   "LOW X"
USING (FINDLOWX PYRAMID-1 0 820 0 1200 0) (LOWX 10 PYRAMID-1 0)
  (LOCAT BLOCK-2 400 0 0) (HASSIZE BLOCK-2 200 200 200)
INSERTING (LOWX 10 PYRAMID-1 600) (NOT (LOWX 10 PYRAMID-1 0))

I 106.  Q65-2   "LOW X"
USING (FINDLOWX PYRAMID-1 0 820 0 1200 0) (LOWX 10 PYRAMID-1 0)
  (LOCAT BLOCK-5 300 640 0) (HASSIZE BLOCK-5 300 100 400)
WARNING (10 PYRAMID-1 600) ALREADY UNDER LOWX  =.
WARNING (10 PYRAMID-1 0) NOT UNDER LOWX  =-
INSERTING (LOWX 10 PYRAMID-1 600) (NOT (LOWX 10 PYRAMID-1 0))   Q65Q65Q65/Q66/

I 107.  Q66-1   "LOW Y"
USING (FINDLOWY PYRAMID-1 0 1200 0 373 0) (LOWY 10 PYRAMID-1 0)
  (LOCAT BLOCK-1 100 100 0) (HASSIZE BLOCK-1 100 100 100)
INSERTING (LOWY 10 PYRAMID-1 200) (NOT (LOWY 10 PYRAMID-1 0))

I 108.  Q66-2   "LOW Y"
USING (FINDLOWY PYRAMID-1 0 1200 0 373 0) (LOWY 10 PYRAMID-1 0)
  (LOCAT BLOCK-2 400 0 0) (HASSIZE BLOCK-2 200 200 200)
WARNING (10 PYRAMID-1 200) ALREADY UNDER LOWY  =.
WARNING (10 PYRAMID-1 0) NOT UNDER LOWY  =-
INSERTING (LOWY 10 PYRAMID-1 200) (NOT (LOWY 10 PYRAMID-1 0))   Q65Q66Q66/Q67/

I 109.  Q67-1   "GROW READY"
USING (GROWTOFITO 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)
INSERTING (GROWTOFIT 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)
  (CHECKFAILFIT 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)
  (NOT (GROWTOFITO 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100))   Q68/

I 110.  Q68-1   "SIZES FIT"
USING (GROWTOFIT 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)
  (LOWX 10 PYRAMID-1 600) (LOWY 10 PYRAMID-1 200)
  (FINDLOWX PYRAMID-1 0 820 0 1200 0) (FINDLOWY PYRAMID-1 0 1200 0 373 0)
INSERTING (FINDHIGHX PYRAMID-1 700 1200 200 1200 0)
  (FINDHIGHY PYRAMID-1 600 1200 300 1200 0)
  (FOUNDHIGHPAIRO 10 PYRAMID-1 600 200 0) (HIGHX 10 PYRAMID-1 1200)
  (HIGHY 10 PYRAMID-1 1200)
  (NOT (GROWTOFIT 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100))
  (NOT (LOWX 10 PYRAMID-1 600)) (NOT (LOWY 10 PYRAMID-1 200))
  (NOT (FINDLOWX PYRAMID-1 0 820 0 1200 0))
  (NOT (FINDLOWY PYRAMID-1 0 1200 0 373 0))
  (NOT (CHECKFAILFIT 10 PYRAMID-1 0 0 1200 1200 0 820 373 100 100 100)) Q73Q70/
  Q71/

I 111.  Q71-1   "HIGH Y"
USING (FINDHIGHY PYRAMID-1 600 1200 300 1200 0) (HIGHY 10 PYRAMID-1 1200)
  (LOCAT BOX-1 600 600 0) (HASSIZE BOX-1 600 600 1)
INSERTING (HIGHY 10 PYRAMID-1 600) (NOT (HIGHY 10 PYRAMID-1 1200))      Q73Q71/
  Q72/

I 112.  Q72-1   "HIGH READY"
USING (FOUNDHIGHPAIRO 10 PYRAMID-1 600 200 0)
INSERTING (FOUNDHIGHPAIR 10 PYRAMID-1 600 200 0)
  (NOT (FOUNDHIGHPAIRO 10 PYRAMID-1 600 200 0)) Q73/

I 113.  Q73-1   "HIGH PAIR"
USING (FOUNDHIGHPAIR 10 PYRAMID-1 600 200 0) (HIGHX 10 PYRAMID-1 1200)
```

```
  (HIGHY 10 PYRAMID-1 800) (FINDHIGHX PYRAMID-1 700 1200 200 1200 0)
  (FINDHIGHY PYRAMID-1 800 1200 300 1200 0)

TRACING
  FOUND REGION (800 200 0) TO (1200 800 0)


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (LOCATERESULT PYRAMID-1 800 200 1200 800 0)
  (NOT (FOUNDHIGHPAIR 10 PYRAMID-1 800 200 0)) (NOT (HIGHX 10 PYRAMID-1 1200))
  (NOT (HIGHY 10 PYRAMID-1 600)) (NOT (FINDHIGHX PYRAMID-1 700 1200 200 1200 0))
  (NOT (FINDHIGHY PYRAMID-1 800 1200 300 1200 0)) (TRACING T)  Q78/Q77/Q78/

1114. Q78-1  "LOCATE RANDOM"
USING (LOCATERESULT PYRAMID-1 800 200 1200 800 0)
  (USERESULT TABLE-1 PYRAMID-1 100 100 RANDOM)
INSERTING (FOUNDSPACE TABLE-1 PYRAMID-1 1072 458 0)
  (NOT (LOCATERESULT PYRAMID-1 800 200 1200 800 0))
  (NOT (USERESULT TABLE-1 PYRAMID-1 100 100 RANDOM))   W53W53AW54W54AW12/

1115. W12-1  "GET RID FND"
USING (GETRIDPUT G-2 PYRAMID-1 TABLE-1)
  (FOUNDSPACE TABLE-1 PYRAMID-1 1072 458 0) (HASLEVEL G-2 2) (CHOICECOUNT 0)
  (EVENTTIME 0)

TRACING
..GOAL G-3 PUT PYRAMID-1 (1072 458 0)


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (PUT G-3 PYRAMID-1 1072 458 0) (HASLEVEL G-3 3) (HASSUPERGOAL G-3 G-2)
  (NOT (GETRIDPUT G-2 PYRAMID-1 TABLE-1))
  (NOT (FOUNDSPACE TABLE-1 PYRAMID-1 1072 458 0)) (NOT (CHOICECOUNT 0))
  (CHOICECOUNT 1) (GETRIDCHOICE 1 G-3 1 TABLE-1 PYRAMID-1 1072 458 0)
  (CHOICETIME 1 0) (TRACING T) Q31/

1116. Q31-1  "PUT"
USING (PUT G-3 PYRAMID-1 1072 458 0) (HASLEVEL G-3 3)

TRACING
...GOAL G-4 GRASP PYRAMID-1


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (GRASP G-4 PYRAMID-1) (NEXT G-4 (PUTMOVE G-3 PYRAMID-1 1072 458 0))
  (HASLEVEL G-4 4) (NOT (PUT G-3 PYRAMID-1 1072 458 0)) (TRACING T)    Q43Q45/

1117. Q45-1  "GRASP"
USING (GRASP G-4 PYRAMID-1) (LOCAT PYRAMID-1 100 100 100)
  (HASSIZE PYRAMID-1 100 100 100) (HASLEVEL G-4 4)

TRACING
....GOAL G-5 CLEAROFF PYRAMID-1


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (CLEAROFF G-5 PYRAMID-1) (NEXT G-5 (GRASP1 G-4 PYRAMID-1 150 150 200))
  (HASLEVEL G-5 5) (NOT (GRASP G-4 PYRAMID-1)) (TRACING T)     W4/W3/W6/

1118. W6-1  "CLEAR ."
USING (CLEAROFF G-5 PYRAMID-1) (CLEARTOP PYRAMID-1)
INSERTING (SUCCEED G-5) (NOT (CLEAROFF G-5 PYRAMID-1))  W0/

1119. W0-1  "SUCC NEXT"
USING (SUCCEED G-5) (NEXT G-5 (GRASP1 G-4 PYRAMID-1 150 150 200))
  (HASLEVEL G-5 5)

TRACING
       G-5 SUCCEEDS


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (GRASP1 G-4 PYRAMID-1 150 150 200) (NOT (SUCCEED G-5)) (TRACING T)
  Q46/

1120. Q46-1  "GRASP MOVE"
USING (GRASP1 G-4 PYRAMID-1 150 150 200)
INSERTING (MOVEHAND 150 150 200) (GRASP2 G-4 PYRAMID-1)
  (NOT (GRASP1 G-4 PYRAMID-1 150 150 200))     Q3/Q2LQ2Q1/

1121. Q1-1  "MOVE HAND"
USING (MOVEHAND 150 150 200) (ISA HAND-1 HAND) (LOCAT HAND-1 0 100 400)
  (EVENTTIME 0)
```

```
TRACING
     (0) MOVING HAND FROM (0 100 400) TO (150 150 200)


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (LOCAT HAND-1 150 150 200) (NOT (MOVEHAND 150 150 200))
  (NOT (LOCAT HAND-1 0 100 400)) (NOT (EVENTTIME 0)) (EVENTTIME 1)
  (UNEVENT 0 (MOVEHAND 0 100 400)) (TRACING T)  F81F82F83F84F85F86Q49Q39Q70Q69
  Q2LQ2Q1Q71Q70Q66Q65Q64Q61Q57W84V53LV53RV53DW28W22W12W57W58W54W53W45W42W38W
  W34W33W17W20Q49Q47UQ47/

1122. Q47-1  "GRASP ACT"
USING (GRASP2 G-4 PYRAMID-1) (ISA HAND-1 HAND) (EVENTTIME 1)

TRACING
     (1) GRASPING PYRAMID-1


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (SUCCEED G-4) (GRASPING HAND-1 PYRAMID-1) (NOT (GRASP2 G-4 PYRAMID-1))
  (NOT (EVENTTIME 1)) (UNEVENT 1 (UNGRASP PYRAMID-1)) (EVENTTIME 2) (TRACING T)
     W07/W05/W0/

1123. W0-2  "SUCC NEXT"
USING (SUCCEED G-4) (NEXT G-4 (PUTMOVE G-3 PYRAMID-1 1072 458 0))
  (HASLEVEL G-4 4)

TRACING
     G-4 SUCCEEDS


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (PUTMOVE G-3 PYRAMID-1 1072 458 0) (NOT (SUCCEED G-4)) (TRACING T)
  Q32/

1124. Q32-1  "PUT MOVE"
USING (PUTMOVE G-3 PYRAMID-1 1072 458 0) (HASSIZE PYRAMID-1 100 100 100)
INSERTING (MOVEHAND 1122 508 100) (UNGRASP PYRAMID-1) (SUCCEED G-3)
  (NOT (PUTMOVE G-3 PYRAMID-1 1072 458 0))     Q2/

1125. Q2-1  "LIFT OBJECT"
USING (MOVEHAND 1122 508 100) (GRASPING HAND-1 PYRAMID-1)
  (LOCAT PYRAMID-1 100 100 100) (HASSIZE PYRAMID-1 100 100 100)
  (LOCAT HAND-1 150 150 200) (EVENTTIME 2)

TRACING
    (2) LIFTING PYRAMID-1 FROM (100 100 100) TO (1072 458 0)


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (NEWLOCAT PYRAMID-1) (NEWLOCAT2 PYRAMID-1)
  (LOCAT PYRAMID-1 1072 458 0) (TRACING T) (EVENTTIME 3)
  (UNEVENT 2 (MOVEHAND 150 150 200)) (NOT (MOVEHAND 1122 508 100))
  (NOT (LOCAT PYRAMID-1 100 100 100)) (NOT (LOCAT HAND-1 150 150 200))
  (NOT (EVENTTIME 2)) (LOCAT HAND-1 1122 508 100)      Q8/

1126. Q8-1  "REM ON"
USING (NEWLOCAT PYRAMID-1) (LOCAT PYRAMID-1 1072 458 0)
  (HASREL PYRAMID-1 ON BLOCK-1 POS)
INSERTING (REMOHASREL PYRAMID-1 ON BLOCK-1 POS)
  (IRSREMOHASREL PYRAMID-1 ON BLOCK-1 POS) (NOT (NEWLOCAT PYRAMID-1))
  (NOT (HASREL PYRAMID-1 ON BLOCK-1 POS))     Q23/

1127. Q23-1  "OFF CLEAR"
USING (REMOHASREL PYRAMID-1 ON BLOCK-1 POS)
INSERTING (CLEARTOP BLOCK-1)   Q57Q27/Q57FW27PW8Q11/

1128. Q11-1  "OFF STACK"
USING (REMOHASREL PYRAMID-1 ON BLOCK-1 POS) (INSTACK PYRAMID-1 STACK-3)
  (INSTACK BLOCK-1 STACK-3)

TRACING
    TAKING PYRAMID-1 FROM STACK-3


WARNING (T) ALREADY UNDER TRACING *.
INSERTING (REMOINSTACK PYRAMID-1 STACK-3) (NOT (INSTACK PYRAMID-1 STACK-3))
  (TRACING T)  Q13/

1129. Q13-1  "KILL STACK"
USING (REMOINSTACK PYRAMID-1 STACK-3) (INSTACK BLOCK-1 STACK-3)

TRACING
```

```
    STACK-3 DISMANTLED

WARNING (T) ALREADY UNDER TRACING =.
INSERTING (NOT (REMDINSTACK PYRAMID-1 STACK-3)) (NOT (INSTACK BLOCK-1 STACK-3))
 (TRACING T)  Q19/V56FQ29/

! 130. Q29-1  "ERS REM"
USING (ERSREMDHASREL PYRAMID-1 ON BLOCK-1 POS)
INSERTING (NOT (REMDHASREL PYRAMID-1 ON BLOCK-1 POS))
 (NOT (ERSREMDHASREL PYRAMID-1 ON BLOCK-1 POS))      B36F31F34F3418188 11Q7/

! 131. Q7-1   "ADD NEW ON"
USING (NEWLOCAT2 PYRAMID-1) (LOCAT PYRAMID-1 1072 498 0) (LOCAT TABLE-1 0 0 0)
 (ISA TABLE-1 TABLE) (HASSIZE PYRAMID-1 100 100 100)
 (HASSIZE TABLE-1 1200 1700 0)
WARNING (PYRAMID-1) NOT UNDER NEWLOCAT =-
INSERTING (HASREL PYRAMID-1 ON TABLE-1 POS) (NOT (NEWLOCAT2 PYRAMID-1))
 (NOT (NEWLOCAT PYRAMID-1))   B13B31Q81Q82Q49/

! 132. Q49-1  "UNGRASP"
USING (UNGRASP PYRAMID-1) (GRASPING HAND-1 PYRAMID-1)
 (HASREL PYRAMID-1 ON TABLE-1 POS) (EVENTTIME 3)

TRACING
    (3) LETTING GO OF PYRAMID-1


WARNING (T) ALREADY UNDER TRACING =.
INSERTING (NOT (UNGRASP PYRAMID-1)) (NOT (GRASPING HAND-1 PYRAMID-1))
 (NOT (EVENTTIME 3)) (TRACING T) (UNEVENT 3 (GRASP3 HAND-1 PYRAMID-1))
 (EVENTTIME 4) W32W31W43W46W33W34W35W36W42W45W53W54W56W57W12W22W25W17Q2Q1W3
 Q49Q47W20Q47UW22BW58W52BW51W42BQ27/Q21/Q17/Q15/Q6W4W3E12/

! 133. E12-1  "TRACE REL"
USING (HASREL PYRAMID-1 ON TABLE-1 POS)

TRACING
    ADDING PYRAMID-1 ON TABLE-1 (POS)


WARNING (T) ALREADY UNDER TRACING =.
INSERTING (TRACING T)  F34B34B33B10B15B10L/B10K/B10J/V53DV52AV510V31V30V18V17
 F61F62F63F64F65F66M84Q2LQ45Q35Q3Q71Q70Q66Q65Q64Q61V53LV53RW08/

! 134. W0S-1  "SUCC SUPER"
USING (SUCCEED G-3) (HASSUPERGOAL G-3 G-2) (HASLEVEL G-3 3)

TRACING
    G-3 SUCCEEDS


WARNING (T) ALREADY UNDER TRACING =.
INSERTING (SUCCEED G-2) (TRACING T) (NOT (SUCCEED G-3)) W0/

! 135. W0-3   "SUCC NEXT"
USING (SUCCEED G-2) (NEXT G-2 (CLEAROFF G-1 BLOCK-1)) (HASLEVEL G-2 2)

TRACING
   G-2 SUCCEEDS


WARNING (T) ALREADY UNDER TRACING =.
INSERTING (CLEAROFF G-1 BLOCK-1) (NOT (SUCCEED G-2)) (TRACING T)      W3/W4/
 W8/

! 136. W8-2   "CLEAR ."
USING (CLEAROFF G-1 BLOCK-1) (CLEARTOP BLOCK-1)
INSERTING (SUCCEED G-1) (NOT (CLEAROF^ G-1 BLOCK-1))   W0S/W0T/W0/

! 137. W0-4   "SUCC NEXT"
USING (SUCCEED G-1) (NEXT G-1 (FINDSPACE BLOCK-5 BLOCK-1 100 100 100))
 (HASLEVEL G-1 1)

TRACING
 G-1 SUCCEEDS


WARNING (T) ALREADY UNDER TRACING =.
INSERTING (FINDSPACE BLOCK-5 BLOCK-1 100 100 100) (NOT (SUCCEED G-1))
 (TRACING T)  Q51/

TRACING
```

```
  G-6 SUCCEEDS

WARNING (T) ALREADY UNDER TRACING =.
INSERTING (SUCCEED GT) (TRACING T) (NOT (SUCCEED G-6))  W0/W0S/W0T/

! 161. W0T-1  "SUCC TOP"
USING (SUCCEED GT) (HASLEVEL GT 0)

TRACING
GT SUCCEEDS


WARNING (T) ALREADY UNDER TRACING =.
INSERTING (TRACING T) (NOT (SUCCEED GT)) (NOT (HASLEVEL GT 0))  Q43W0FW28XW28
 W18W140W14W54XW24FW24W23FW10W1W0GW38W54AW53AQ31W28DW23BW18W18W19W13W
 W28ZW18EV52/

! 162. V52-1  "CHECK PUTON"
USING (CHECKPUTON BLOCK-1 ON BLOCK-5)
INSERTING (CHECKPUTON2 BLOCK-1 ON BLOCK-5) (NOT (CHECKPUTON BLOCK-1 ON BLOCK-5))
     V52T/V52A/

! 163. V52A-1  "PUTON OK"
USING (CHECKPUTON2 BLOCK-1 ON BLOCK-5) (HASREL BLOCK-1 ON BLOCK-5 POS)
INSERTING (REPLY0 (OKAY)) (NOT (CHECKPUTON2 BLOCK-1 ON BLOCK-5))       V0/

! 164. V0-1   "COUNT REPLY"
USING (REPLY0 (OKAY)) (NREPLY 0)
INSERTING (REPLY 1 (OKAY)) (NREPLY 1) (NOT (REPLY0 (OKAY))) (NOT (NREPLY 0))
 V5V15V0M82PM83M83PM84FM85M86Y23V18V12V10B53/

! 165. B53-1  "NPBND UNDOP"
USING (NPBOUND RE-1) (CUROBJP OBJ-2 OBJ-1) (REFERS OBJ-2 BLOCK-5)
INSERTING (NOT (CUROBJP OBJ-2 OBJ-1))  G18G17N18N15

2

REPLY (1 (OKAY))
```

DATE
FILMED
— 8